昭和63年 7月 1 日発行（毎月 1 回 1 日発行）第 7 巻 7 号通巻75号 昭和58年11月 2 日第三種郵便物認可

ISSN 0910-7614

パーソナルコンピュータ・マガジン
MZシリーズ,X1/turbo,X68000&ポケコン

Oh!X

特集
実践C言語からの誘惑

マイコンショウ/ビジネスショウレポート'88
速報OS-9/X68000



新連載 C調言語講座PRO-68K
まずはprintfより始めよ

X68000 BASIC入門
無限作曲・MML伝説

オブジェクト指向のゲームプログラミング
完結のスネークオブジェクト

S-OS 全機種共通システム
ウィンドウ用ドライバMW-1
構造化言語SLANG入門(2)

ミュージックプログラムLIVE in '88
MZ-2500TRUTH/MZ-1500テクノポリス
X1/turboアフターバーナー

熱烈歓迎！ 話題のゲームソフトレビュー
ソーサリアン/SUPER大戦略
ゼリアード/アルギースの翼
最新3大麻雀ソフトの饗宴

知能機械概論/Between The Lines
祝一平の「人類タコ科図鑑」

7
JUL.1988
定価540円



ますます熱くなる。
クリエイティブワークステーションX68000。

●新たなゆとりが創造力を刺激する——。20MバイトハードディスクをX68000本体に内蔵したX68000 ACE HD の登場です。もちろん、X68000としての本質は変わるはずもなく、あのクリエイティブなX68000そのものです。といって、たとえ3.5インチのハードディスクとはいえ、それをスリムなマンハッタンシェイプの本体内に搭載するには、これまで以上の実装密度が要求されます。このハードディスクモデルには、集積度をさらに高めたカスタムICや、メモリとして1MビットのダイナミックRAMが採用されていますが、これは、いわば過去1年間の成果というべきもので、ある意味では、ビジュアルシェルなどのソフトウェアに対してハードウェアのユーザーインターフェイスとも言えるでしょう。

●約110本、X68000のパフォーマンスにふさわしいさまざまなジャンルのソフトウェアがすでに流通。このマシンのソフト環境は着実な歩みを見せています。この間、ユーザー各位の熱烈なご支持とシステムハウス各位の開発ご努力に心からの感謝をささげるとともに、そうしたご厚意に対して、私たちは将来的な展望も含めて、でき得るかぎりのサポートをお約束するものです。

〈X68000ACE HD の主な特長〉●3.5インチ20Mバイトタイプのハードディスク(平均アクセスタイム80ms)を1基内蔵 ●実装密度を追求して信頼性を高めたマンハッタンシェイプ ●68000搭載 ●テキスト、グラフィック、スプライト、独立3画面設計、最大12Mバイトの大容量メモリ(標準1Mバイト) ●フレンドリーOS、Human 68k搭載 ●連文節変換、マルチフォントをサポートした強力日本語処理 ●1024×1024ドット(最大表示エリア768×512ドット)の実画面エリアを装備した高解像度表示能力 ●512×512ドット、65,536色同時発色 ●水平32、1画面128のパワフルなスプライト機能 ●オーバースキャン機能を採用した512×512ドットレベルのスーパーインポーズ ●テキストビットマップ方式採用 ●8重和音ステレオFM音源搭載 ●音声デジタイズ記録AD PCM※採用 ●マウス・トラックボール標準装備 ●5インチ1MバイトFDD2基搭載 ●「X-BASIC」、「辞書ディスク」と各種ユーティリティ、「日本語ワードプロセッサ」をバンドル

※Adaptive Differential PCM

豊富な周辺機器がクリエイティブワークをサポート。

●15型カラーディスプレイ	CU-15M1-E	標準価格 99,800円
●カラーイメージスキャナ※1	CZ-8NS1	標準価格188,000円
●カラーイメージユニット※2	CZ-6VT1	標準価格 69,800円
●カラービデオプリンタ	CZ-6PV1	標準価格198,000円
●24ピン漢字プリンタ(80桁)	CZ-8PK7	標準価格122,000円
●24ピン漢字プリンタ(136桁)	CZ-8PK8	標準価格152,000円
●24ピン漢字プリンタ(80桁)	CZ-8PK9	標準価格 89,800円
●熱転写カラー漢字プリンタ	CZ-8PC2	標準価格 69,800円
●ハードディスクユニット(20MB)	CZ-620H	標準価格178,000円
●モデムユニット※3	CZ-8TM2	標準価格 49,800円
●RS-232Cケーブル(平行接続型)	CZ-8LM1	標準価格 7,200円
●RS-232Cケーブル(クロス接続型)	CZ-8LM2	標準価格 7,200円
●拡張I/Oボックス(4スロット)	CZ-6EB1	標準価格 88,000円
●1MB増設RAMボード(内蔵用)	CZ-6BE1A	標準価格 38,000円
●2MB増設RAMボード※4	CZ-6BE2	標準価格 79,800円
●4MB増設RAMボード※4	CZ-6BE4	標準価格138,000円
●GP-IBボード	CZ-6BG1	標準価格 59,800円
●ユニバーサルI/Oボード	CZ-6BU1	標準価格 39,800円
●増設用RS-232Cボード(2チャンネル)	CZ-6BF1	標準価格 49,800円
●数値演算プロセッサボード	CZ-6BP1	標準価格 79,800円
●スキャナ用パラレルボード	CZ-6BN1	標準価格 29,800円
●システムラック	CZ-6SD1	標準価格 44,800円
●アンプ内蔵スピーカーシステム(2本1組)	AN-160SP	標準価格 59,800円
●ジョイカード	CZ-8NJ1	標準価格 1,700円

※1 使用に際しては、カラーイメージスキャナCZ-8NS1に同梱のRS-232Cケーブルで接続するか、より高速のパラレルデータ転送を行う場合、別売のスキャナ用パラレルボードCZ-6BN1で接続してください。※2 使用に際してはコンピュータ本体と専用15型カラーディスプレイテレビ(CZ-601D、CZ-611Dなど)が必要です。※3 モデムユニットCZ-8TM2に同梱のソフトはX1/X1 turboシリーズ用です。※4 使用に際しては、あらかじめ、別売の1MB増設RAMボードCZ-6BE1Aを増設してください。

アートツールと呼びたい「PRO-68K」シリーズソフト。

MUSIC PRO-68K CZ-213MS 標準価格18,800円
メロディ譜、ピアノ譜、最大8パートのスコア(総譜)を自由なレイアウトで書き込んだ譜面を内蔵のFM音源で演奏できる楽譜ワープロ&演奏用ミュージックツール。

SOUND PRO-68K CZ-214MS 標準価格15,800円
FM音源のパラメータを直接指定したり、エンベロープやビブラートを言葉による音のイメージ指定で思いどおりの音色が作成できるサウンドエディティングツール。

BUSINESS PRO-68K CZ-212BS 標準価格68,000円
スプレッドシート、データベース、グラフ作成機能を緊密に一体化させた統合ビジネスツール。マウス対応のイージーオペレーション、最大16個のマルチウインドウが使えます。

C compiler PRO-68K CZ-211LS 標準価格39,800円
Cコンパイラ、BASIC-Cコンバータ、アセンブラ、リンカ、デバッガ、アーカイバ、コンバータで構成。Human 68k上におけるプログラム開発を効率良くサポートします。

〈ゲームソフト〉 ●ツインビー CZ-217AS 標準価格7,800円
●アルカノイド CZ-222AS 標準価格7,800円

さらに洗練されて信頼性を高めた
ハイコストパフォーマンスFDモデルX68000ACE

X68000
PERSONAL WORKSTATION
ACE



■本体+キーボード
CZ-601C-GY(グレー)・-BK(ブラック) 標準価格319,800円

写真の15型カラーディスプレイテレビCZ-601D-GY(グレー)・-BK(ブラック)、チルトスタンドCZ-6ST1-E(グレー)・-B(ブラック)は別売です。

〈パソコン教室開催のお知らせ〉 X68000、MZ-2861のパソコン教室を開催します。くわしくは、下記までお問い合せください。
札幌(011)642-8111・仙台(022)288-8705・東京(03)260-1161・横浜(045)201-6525・名古屋(052)332-2611・大阪(06)222-7655・神戸(078)291-8715・福岡(092)481-2860



表紙絵：Matsubaguchi Tadao

■広告目次

Oh!X

C O N T

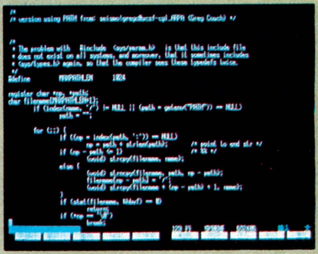
●特集

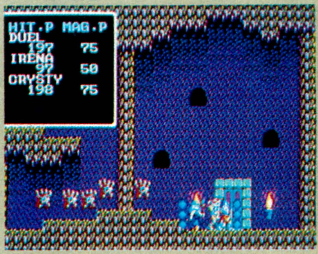
〈スタッフ〉
●編集長／前田 徹 ●副編集長／永野 仁 ●編集／植木章夫 石塚康世 高野庸一 ●協力／有田隆也 中森 章 清水和人 後藤貴行 林 一樹 浅野恵造 山村 一 井本 泰 山田伸一郎 堀内保秀 荻窪 圭 藤原和典 岡本浩一郎 毛内俊行 野中俊一郎 吉田賢司 影山裕昭 相馬英智 古村 聡 村田敏幸 ●カメラ／杉山和美 ●イラスト／永沢しげる 山田晴久 小栗由香 ●アートディレクター／島村勝頼 ●レイアウト／元木昌子 AD GREEN ●校正／手塚喜美子 千野延明

1988 JUL. 7

E N T S

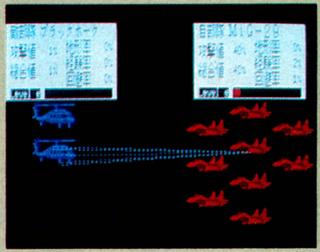

マイコンショウ/ビジネスショウレポート'88

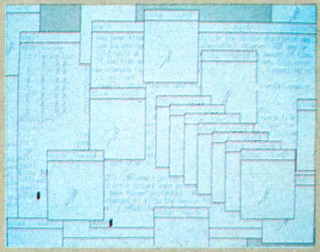

特集 実践C言語からの誘惑



ソーサリアン

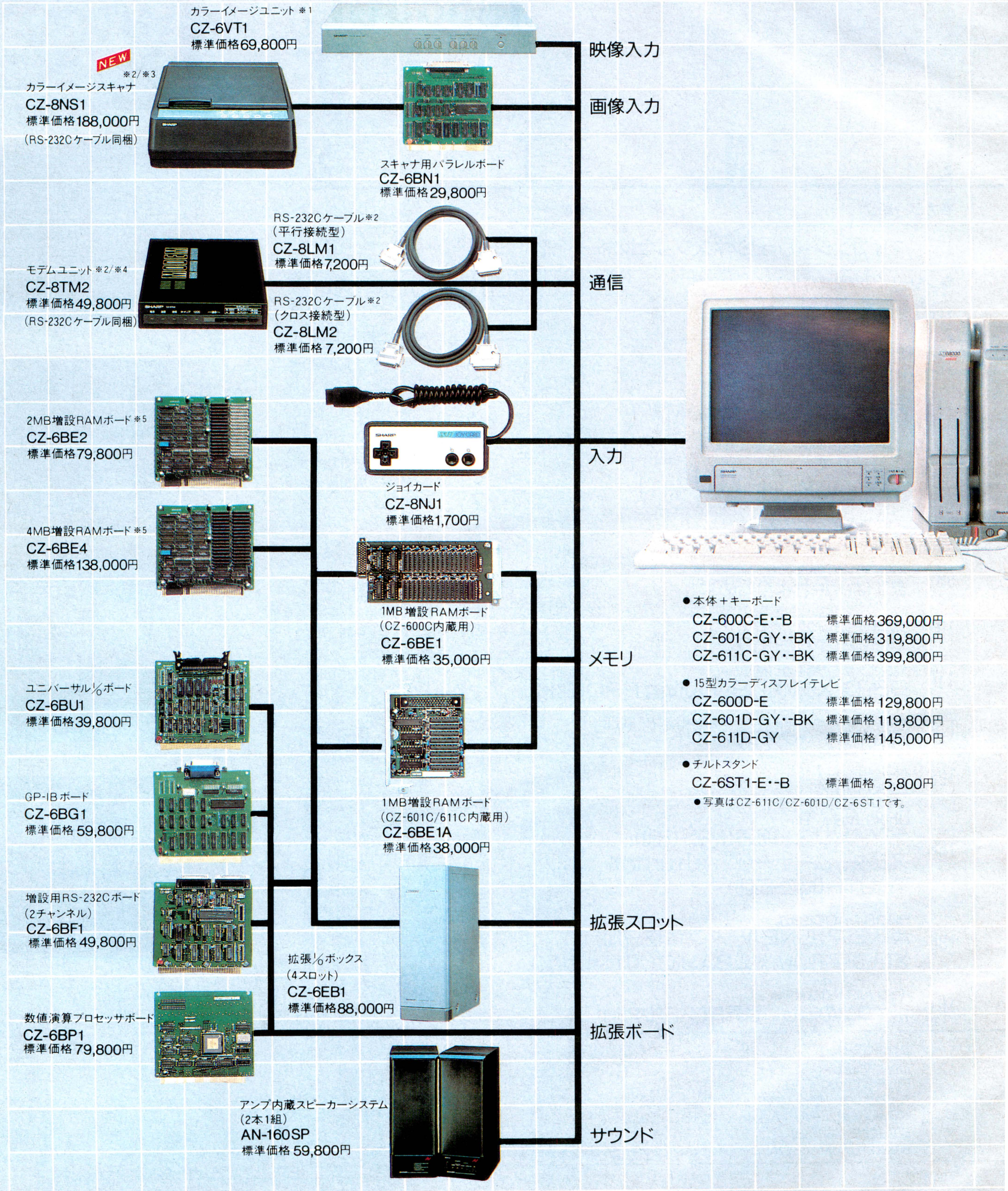

ゼリアード


SUPER大戦略


X68000あなたの知らない世界

SHARP

クリエイティブマインド



※1使用に際してはコンピュータ本体と専用15型カラーディスプレイ(CZ-601D、CZ-611Dなど)が必要です。※2 X1/X1ターボシリーズと共用。※3ご使用に際しては、カラーイメージスキャナ CZ-8NS1に同梱のRS-232Cケーブ
さい。※4 モデムユニットCZ-8TM2に同梱のソフトは X1/X1ターボシリーズ用です。※5 使用に際しては、あらかじめ別売の1MB増設RAMボードCZ-6BE1 標準価格35,000円(CZ-600C)、CZ-6BE1A 標準価格38,000

シャープ株式会社 ●お問い合わせは…シャープ㈱電子機器事業本部システム機器営業部〒545 大阪市阿倍野区長池町22番22号☎(06)621-1221(大代表)電子機

資料請求券

思わず熱くなる。

ちふれる周辺機器がX68000をサポート。



シャープペリフェラルファミリー

X68000

表示

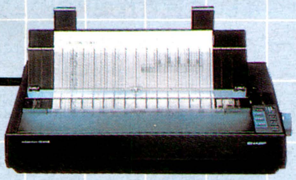


15型カラーディスプレイ※2
CU-15M1-E
標準価格99,800円

プリントアウト



24ピン漢字プリンタ(80桁)※2
CZ-8PK7
標準価格122,000円
(信号ケーブル同梱)



24ピン漢字プリンタ(136桁)※2
CZ-8PK8
標準価格152,000円
(信号ケーブル同梱)



24ピン漢字プリンタ(80桁)※2
CZ-8PK9
標準価格89,800円
(信号ケーブル同梱)



熱転写カラー漢字プリンタ※2
CZ-8PC2
標準価格69,800円
(信号ケーブル同梱)

ビデオプリント



カラービデオプリンタ※2
CZ-6PV1
標準価格198,000円
(信号ケーブル同梱)

ハードディスク



ハードディスクユニット(20MB)
CZ-620H
標準価格178,000円

システムラック



システムラック
CZ-6SD1
標準価格44,800円

接続するか、より高速のパラレルデータ伝送を行う場合、別売のスキャナ用パラレルボードCZ-6BN1 標準価格29,800円で接続してくだ
CZ-601C、CZ-611C)を増設してください。

X1ファミリーの システムづくりに応える 多彩な周辺機器群

映像編集装置		
●カラーイメージスキャナ	CZ-8NS1	188,000円
●カラーイメージボードII	CZ-8BV2	39,800円
●立体映像セット	CZ-8BR1	29,800円
●パーソナルテロッパ※1	CZ-8DT2	44,800円
プリンタ		
●24ピン漢字プリンタ(80桁)	CZ-8PK5	129,000円
●24ピン漢字プリンタ(136桁)	CZ-8PK6	159,000円
●ドットプリンタ	CZ-8PD3	59,800円
FM音源		
●ステレオタイプFM音源ボード	CZ-8BS1	23,800円
※スピーカー(2本1組)標準装備、ミュージックツール同梱		
ファイル装置		
●ミニフロッピーディスクユニット(2HD・2D)※2	CZ-520F	118,000円
●ミニフロッピーディスクユニット(2D)	CZ-502F	99,800円
●ミニフロッピーディスクユニット(2D・1ドライブ)	CZ-503F	49,800円
●ハードディスクユニット(10MB)	CZ-500H	348,000円
●増設用ハードディスクユニット(10MB)	CZ-501H	258,000円
●カセットデータレコーダ	CZ-8RL1	24,800円
●ミニフロッピーディスク	CZ-5M2D/CZ-5M2HD	(各10枚入)
●コンパクトフロッピーディスク	CZ-3FBD	1,300円
拡張ボード・その他		
●320KB外部メモリ	CZ-8BE2	29,800円
●RS-232C・マウスボード※3	CZ-8BM2	19,800円
●JIS第1水準漢字ROM※4	CZ-8BK2	19,800円
●JIS第2水準漢字ROM※5	CZ-8BK4	6,800円
●JIS第2水準漢字ROM&ターボ博士レキシコン・日本語百科ワードパワー※6	CZ-8BK3	13,800円
●フロッピーディスクインターフェイス※7	CZ-8BF1	14,800円
●RS-232C用ケーブル(平行接続型)	CZ-8LM1	7,200円
●RS-232C用ケーブル(クロス接続型)	CZ-8LM2	7,200円
●拡張I/Oポート※8	CZ-8EP	11,800円
●拡張I/Oボックス	CZ-8EB3	33,800円
●RFコンバータ※9	AN-58C	2,980円
●モデムユニット(300ボー)	CZ-8TM1	29,800円
●モデムユニット(300/1200ボー自動切換)	CZ-8TM2	49,800円
●マウス	CZ-8NM2	6,800円
●チルトスタンド※10	CZ-6ST1(E・B)	5,800円
●チルトスタンド※11	CZ-81T(S・R)	8,500円
●システムスタンド	CZ-8SS2	5,500円
●ジョイカード	CZ-8NJ1	1,700円
●スキャナ用パラレルボード※12	CZ-8BN1	27,800円

(価格は標準価格です。)

●品番中の( )表示は、S〈メタリックシルバー〉・R〈ローズレッド〉・E〈オフィスグレー〉・B〈ブラック〉を示します。※1 CZ-862Cには接続できません ※2 X1ターボシリーズ用 ※3 X1シリーズ用 ※4 CZ-800C、801C、802C、803C、811C、820C用 ※5 CZ-856C用 ※6 CZ-850C、851C、852C、862C用 ※7 CZ-850CでCZ-520Fを使用する場合、またCZ-803C、804C、811C、820C、850CでCZ-300Fを使用する場合に必要 ※8 CZ-800C、802C用 ※9 CZ-820C、822C、830C用 ※10 CZ-600D、601D、611D、880D、830D、CU-15M1用 ※11 CZ-801D、802D、811D、850D、855D、870D用 ※12 CZ-8NS1用 ●接続等の詳細につきましては、周辺機器総合カタログをご参照ください。

事業本部テレビ事業部第4商品企画部〒162 東京都新宿区市谷八幡町8番地☎(03)260-1161(大代表)

SHARP

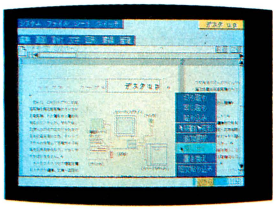
MZの新しいソフト環境

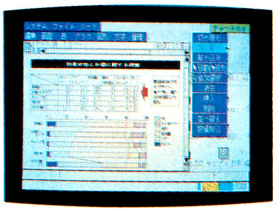
日本語ワードプロセッサ「書院28」の搭載、「MS-DOS™V3.1」の標準装備、
市販アプリケーション活用のための「エミュレーションソフト」の搭載…
数々のソフトウェア上の特長を持つMZ-2861に、いま新たなシステム展開。

OAソフトウェア UPシリーズ

これからの企画書、提案書作りに新しいOAツール。

MZ-2861の日本語入力機能を有機的に活かす統合OAソフトウェア「UPシリーズ」の登場です。**デスクトップパブリッシング**という新しいジャンルのレイアウトワープロ、**集計表・グラフ作成統合**ソフトウェア、自由度の高い**カード型データベース**、アウトラインプロセッサというジャンルの新しい**企画書作成**ソフトウェア…。オフィスワークを代表的な4つの局面からアプローチして専門化したOAツールです。「パソコンファクス28」とのリンクも可能。





日本語レイアウトワープロ■**デスクUP(IP-1251)**	標準価格88,000円
集計表・グラフ作成ソフト■**チャートUP(IP-1252)**	標準価格55,000円
カード型データベース■**UPクリッパー(IP-1253)**	標準価格77,000円
企画書作成ソフト■**プランUP(IP-1254)**	標準価格66,000円

イメージ処理ツール

絵や写真を取り込んで多彩に処理。

この「ハンディ・COPY KIT」や「COLOR IMAGE EDITOR」、「ハンディカラースキャナ」は、絵や写真をコンピュータのイメージデータとして手軽に取り込み、編集・活用するためのツールです。取り込んだデータは、統合化ソフトやワープロソフトなど他のアプリケーションとの連携で応用範囲もさらに広がります。



ハンディ・COPY KIT SS-SC28M 標準価格49,800円

モノクロハンディスキャナと、カラー処理もできるカラーイメージエディタを組み合わせたキットです。デスクトップパブリッシングへの活用など、イメージ処理により表現力がさらにアップします。

●専用シリアルインターフェイスボード、ACアダプタ同梱。



モノクロハンディスキャナ(同梱)

COLOR IMAGE EDITOR SS-SC28C 標準価格29,800円

ハンディカラースキャナ(WD-05HS・別売)をMZ-2861の環境のもとで活用するためのスキャナ用カラー画像取り込み処理ツールです。

●専用シリアルインターフェイスボード、ACアダプタ同梱。

■**ハンディカラースキャナ** WD-05HS 標準価格49,800円



ハンディカラースキャナ(別売)

●使用プリンタ：モノクロ/MZ-1P17、MZ-1P18、MZ-1P19、MZ-1P27、MZ-1P28、MZ-1P29 カラー/IO-725 ●データを利用できるアプリケーション：デスクUP(IP-1251)、チャートUP(IP-1252)、UPクリッパー(IP-1253)、プランUP(IP-1254)、パソコンファクス28(IP-1256)、一太郎Ver2.1※、花子※

※㈱ジャストシステム製、またこのソフトを利用するにはMZ-2861本体付属のエミュレーションソフト(V2.0)が必要です。



16ビットパーソナルコンピュータ

MZ-2861

標準価格328,000円

●写真の14型カラーディスプレイMZ-1D26標準価格89,800円マウスMZ-1X29標準価格13,800円は別売。

アプリケーションと有機的にリンクする日本語環境

●連文節変換サポート、JIS第1/第2水準漢字ROMはもちろん、約10万語(内9万語はROM)の辞書を内蔵した高機能日本語ワードプロセッサ「書院28」の搭載。またMS-DOS上のアプリケーションで「書院28」と同等の日本語入力が行なえるフロントエンドプロセッサで、ビジネスワープロとMS-DOSが融合したフレンドリーな実務環境を実現しました。

●レーザープリンタ MZ-1P23 950,000円／●漢字水平インサータプリンタ MZ-1P27 268,000円／●80桁漢字プリンタ MZ-1P28 148,000円／●136桁漢字プリンタ MZ-1P29 168,000円／●80桁カラー漢字サーマルプリンタ MZ-1P17(B) 79,800円／●マウス MZ-1X29 13,800円 ※MS-DOSは米国マイクロソフト社の商標です。※価格は標準価格です。

シャープ株式会社
資料のご請求、お問い合わせは…シャープ㈱コンシューマーセンターまで。
西日本OA相談室 〒545 大阪市阿倍野区長池町22番22号 ☎(06)621-1221(大代表)　東日本OA相談室 〒162 東京都新宿区市谷八幡町8番地 ☎(03)260-1161(大代表)

についてお知らせします。

パソコンファクス28

イメージ処理された原稿もダイレクトに鮮明ファクシミリ。



■イメージ情報ステーション
MZ-1V01 標準価格278,000円

イメージ情報ステーションMZ-1V01を使って、「書院28」で作った文書や、イメージ処理された原稿をダイレクトにファクスしたり、受信したファクシミリ原稿を編集して報告書にまとめたりできるコミュニケーションツールです。鮮明、高品位なファクシミリとして注目を集めるパソコンファクスをさらに推し進めたこれからのメディア。UPシリーズ同様に「マルチウインドウ」上で切り換えながら使用でき、一連のUPシリーズソフトウェアとしても活用いただけます。

●パソコンで合成・編集したデータを直接送信●時刻指定同報ファクシミリが可能(最大512ヶ所)●パソコンに直接自動受信可能●原稿の画像をイメージファイルとして取り込み、合成・編集●送信原稿を保存、手軽に呼び出せ、くり返し使用可能●プリンタエミュレーション機能内蔵、市販ソフトをMZ-1V01で印刷、ファクシミリ送信が可能。

■パソコンファクス28 IP-1256 標準価格99,800円

■システム構成

パーソナルコンピュータ	イメージ情報ステーション	アプリケーションソフト	パラレルインターフェイス	マウス	RAMディスク	ハードディスク	MS-DOS	電話機
MZ-2861 (328,000円)	MZ-1V01 (278,000円)	IP-1256 (99,800円)	IP-1256に同梱	MZ-1X29 (13,800円)	任意オプション MZ-1R35 (55,000円)	任意オプション	MZ-2861に標準装備	ファクシミリ機能使用時に市販品をご使用ください。

価格は標準価格です。

エミュレーションソフト

異機種間のソフト利用に新しい概念を導入しました。

全く違うハードウェア間でソフトウェアの互換を持たせる、独創的な発想にもとづいたエミュレーションソフトを標準装備。ひとつのハードウェアに従属するアプリケーションソフトが広く異機種間で使用され、より解放的なソフトウェア環境が期待されます。もちろん、MZ-2861のハードウェア及びBIOSは独自のもの。16ビットパソコンとして数々の特長を装備した上で、付加機能としてエミュレーションソフトをサポートしました。

■エミュレーションソフトV2.0上で動作するPC-98UV2アプリケーション

ジャンル	ソフト名	販売会社
ワープロ	一太郎 VER.2.1	㈱ジャストシステム
	TWINSTAR2 VER.2.00	マイクロプロジャパン㈱
	WORDSTARset VER.3.30C	マイクロプロジャパン㈱
	武蔵98	㈱OAテック
	小次郎98	㈱OAテック
	VJE-Pen	㈱バックス
スクリーンエディタ	MIFES-98 VER.3.0	メガソフト㈱
	RED＋＋ VER.1.27.16	㈱ライフボート

ジャンル	ソフト名	販売会社
表計算/データベース	Super Calc3 Release2 VER.2.07	コンピュータ・アソシエイツ㈱
	Microsoft Multiplan VER.2.01	マイクロソフト㈱
	The CARD2 VER.1.00	㈱アスキー
	LCALC VER.1.1	エイセル㈱
	dBASEIII VER.2.1J	日本アシュトン・テイト㈱
	MIGHTY-BASE II VER.2.0	㈱ソフトウェア・テクノロジー
	Easy File2 VER.2.0C	エー・アイ・ソフト㈱
	創玄 VER.1.00B	エー・アイ・ソフト㈱

ジャンル	ソフト名	販売会社
グラフ/グラフィック	Microsoft CHART VER.2.1	マイクロソフト㈱
	CANDY2 VER.2.3.04	㈱アスキー
	Z's STAFF Kid VER.1.02	㈱アスキー
	花子 VER.1.10	㈱ジャストシステム
	アートマスター400 VER.2.03	㈱システムソフト
ゲーム	上海	㈱システムソフト
	立体版 遊撃王	㈱システムソフト

●現在、当社のテストにより上記23本の動作が確認されていますが、未テストソフトも多数ありますので、この本数はさらに増加するものと思われます。 ●一部ソフトウェアには、動作上、若干の制限事項があります。
●エミュレーションソフトV1.0をお使いの方でMZ-2861ご愛用者カード返送戴いた方にV2.0を無償で贈呈中!

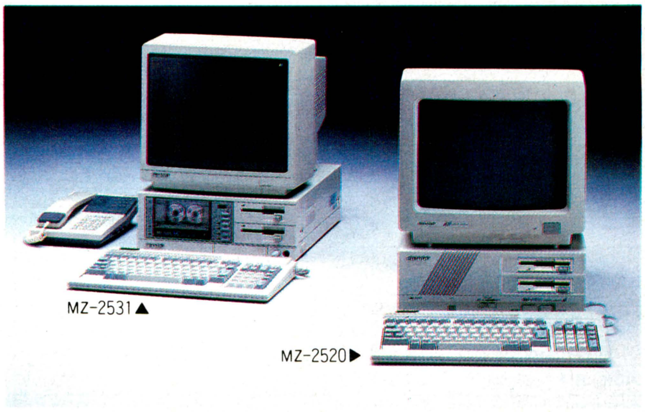
8ビットMZシリーズ

これから始めたい人に……ちょっとぜい沢な入門機。

MZ-2520 標準価格159,800円
※14型カラーディスプレイMZ-1D26標準価格89,800円は別売。

さらにグレードを求める人に……可能性をひろげる高機能。

MZ-2531 標準価格199,800円
※14型カラーディスプレイMZ-1D22標準価格108,000円、モデムホンMZ-1X19は別売。
また装着されているカセットテープは撮影用で、本体の付属品・市販品ではありません。



MZ-2531▲
MZ-2520▶

絵や写真をイメージデータとして手軽に取り込み編集・活用できるイメージ処理ツール

ハンディ・COPY KIT SS-SC25M 標準価格45,000円 ●モノクロハンディスキャナとスキャナ用画像取り込み処理ツールを組み合わせたキット。

カラースキャナ・ユーティリティSS-SC25C 標準価格28,000円 ●ハンディカラースキャナ(WD-05HS標準価格49,800円)をMZ-2500シリーズで活用するためのスキャナ用カラー画像取り込み処理ツール。

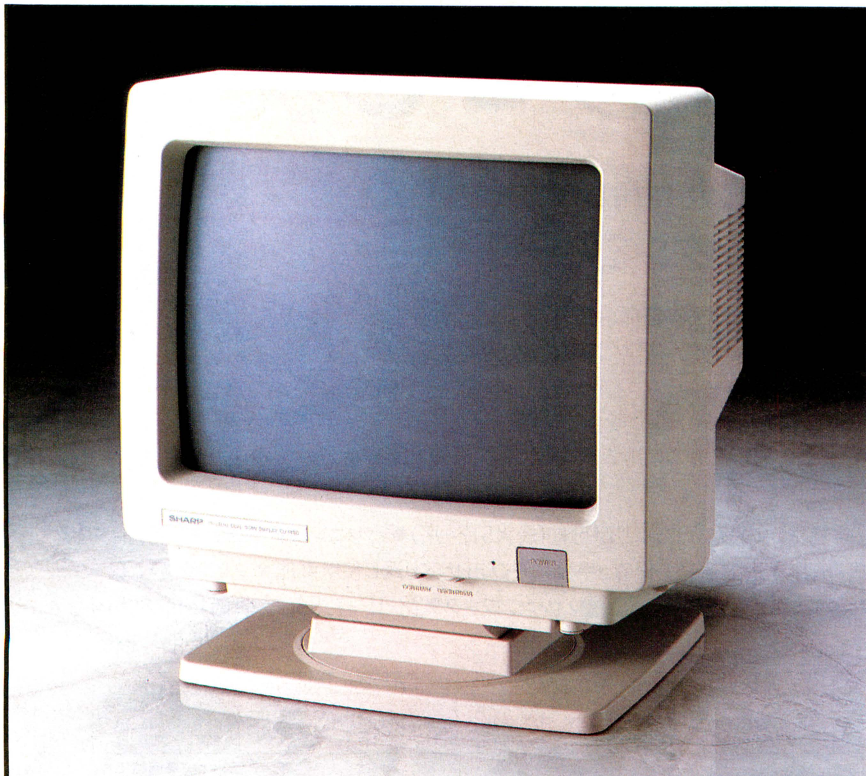
全国のOAショールームにMZ-2500シリーズのソフトを展示中。またMZ-2861、X68000のパソコン教室も開催します。
札幌(011)642-8111/仙台(022)288-8705/東京(03)260-1161/横浜(045)201-6525/名古屋(052)332-2611/大阪(06)222-7655/神戸(078)291-8715/福岡(092)481-2860

資料請求券
MZ-2861
oh! X
7係

SHARP

コンピュータディスプレイ

ここまで身近になった多色化対応。アナログ専用カラーディスプレイ。



●チルトスタンドTF-14AD 標準価格4,800円は別売。

2モードオートスキャン方式採用、鮮やかな65,536色表示、使いやすさを追求したハイコストパフォーマンスモデル。

●入力周波数15/24kHz自動切り換え、2モードオートスキャン方式採用●14型ファインピッチハイコントラストブラウン管採用●アナログRGB専用入力。65,536色などの多色表示が可能●コンテンポラリーな美しいフォルム●コンピュータ接続ケーブル付属●チルトスタンド(別売)装着可能

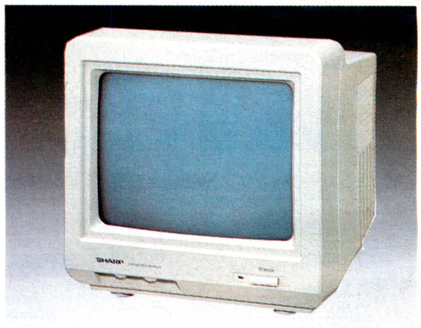
14型カラーディスプレイ
CU-14BD 標準価格 64,800円
アナログ RGB



14 2モードオートスキャン、TVチューナー内蔵
型カラーディスプレイテレビ
CZ-880D(GY・BK) 標準価格109,800円
アナログ/デジタル RGB TV



15 3モードオートスキャン方式採用
型カラーディスプレイ
CU-15M1(E) 標準価格 99,800円
アナログ/デジタル RGB



9 実務レベルに対応する高解像度
型モノクロディスプレイ
MD-9P1 標準価格 34,800円
4050文字 ペーパーホワイト

ディスプレイ \ 仕様		標準価格	サイズ	ブラウン管	ドットピッチ※1 (mm)	表示色数	表示文字数	入力信号方式	備考
多色化対応	CU-14BD	64,800円	14	ファインピッチハイコントラスト	(0.42)	多色※2(アナログ)	実使用4050/2000※3	アナログRGB	Ⓐ Ⓓ
	CU-14AD	84,800円	14	高解像度ハイコントラスト	0.31	多色※2(アナログ)	4050/2000※3	アナログRGB	Ⓐ Ⓓ
	CU-14A4 ★	89,800円	14	高解像度ハイコントラスト	0.39	多色※2(アナログ)/8色(デジタル)	実使用4050	アナログ・デジタルRGB	Ⓐ Ⓓ
	CU-15M1(E)	99,800円	15	高解像度フラットスクエアハイコントラスト	0.39	多色※2(アナログ)/8色(デジタル)	実使用4050/2000※4	アナログ・デジタルRGB	Ⓐ
モノクロ	12M-15B	29,800円	12	高解像度ノングレアハイコントラスト	——	グリーン	2000	コンポジット	Ⓓ
	MD-9P1	34,800円	9	高解像度ノングレアハイコントラスト	——	ペーパーホワイト	4050	コンポジット	Ⓐ Ⓓ
	MD-12P1	39,800円	12	高解像度ノングレアハイコントラスト	——	グリーン	4050	コンポジット	Ⓓ
	MD-12P2	39,800円	12	高解像度ノングレアハイコントラスト	——	ペーパーホワイト	4050	コンポジット	Ⓓ
TVチューナー内蔵	CZ-820D(E)(B)	79,800円	14	ファインピッチハイコントラスト	(0.45)	8色	2000	RGB/コンポジット	Ⓑ
	CZ-830D(BK)	98,000円	14	ファインピッチハイコントラスト	(0.42)	多色※2(アナログ)/8色(デジタル)	実使用4050/2000※3	アナログ・デジタルRGB/コンポジット	Ⓐ Ⓑ Ⓒ
	CZ-880D(GY)(BK)	109,800円	14	高解像度ハイコントラスト	0.31	多色※2(アナログ)/8色(デジタル)	4050/2000※3	アナログ・デジタルRGB/コンポジット	Ⓐ Ⓑ Ⓒ
	CZ-600D(E)(B)★	129,800円	15	高解像度フラットスクエアハイコントラスト	0.39	多色※2(アナログ)/8色(デジタル)	実使用4050/2000※4	アナログ・デジタルRGB/コンポジット	Ⓐ Ⓑ Ⓒ
	CZ-601D(GY)(BK)	119,800円	15	高解像度フラットスクエアハイコントラスト	0.39	多色※2(アナログ)	実使用4050/2000※4	アナログRGB/コンポジット	Ⓐ Ⓑ Ⓒ
	CZ-611D(GY)	145,000円	15	高解像度フラットスクエアハイコントラスト	0.31	多色※2(アナログ)	4050/2000※4	アナログRGB/コンポジット	Ⓐ Ⓑ Ⓒ

●型番中の( )表示は、E/GY〈オフィスグレー〉・B/BK〈ブラック〉を示します。※1( )内はスリットピッチ ※2 512色、4096色、65,536色などコンピュータ出力信号に応じた多色表示が可能 ※3 15kHz/24kHzの自動切換え ※4 15kHz/24kHz/31kHzの自動切換え Ⓐチルトスタンド装着可能(別売) Ⓑデジタルサイン搭載 Ⓒリモコン付 Ⓓ接続ケーブル同梱(CU-14A4、14AD、14BDはアナログ用接続ケーブル付属) ★在庫僅少

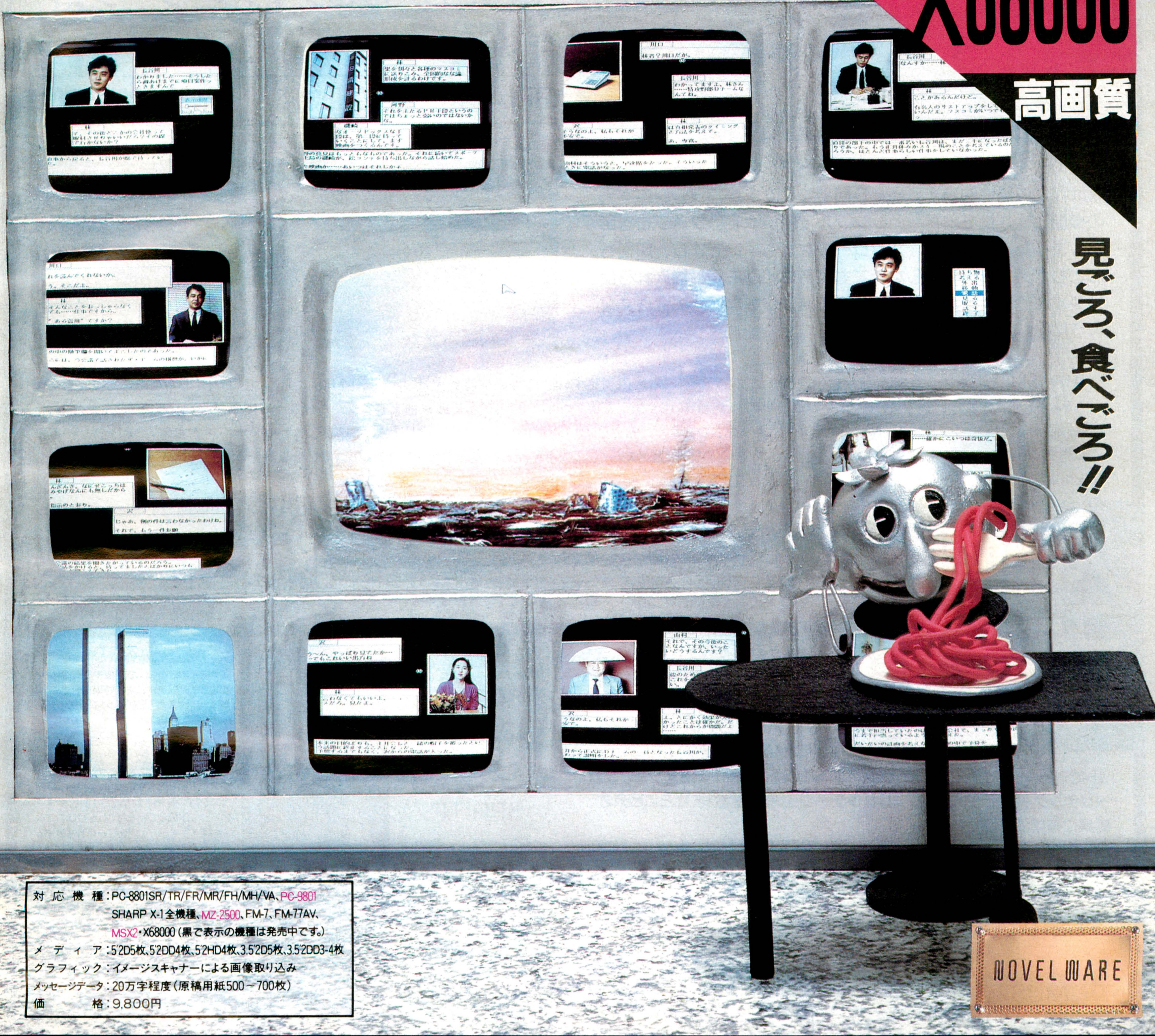
シャープ株式会社 ●お問い合わせは…シャープ㈱電子機器事業本部システム機器営業部〒545 大阪市阿倍野区長池町22番22号☎(06)621-1221(大代表)
電子機器事業本部テレビ事業部第4商品企画部〒162 東京都新宿区市谷八幡町8番地☎(03)260-1161(大代表)

DOME

ドーム/X68000の放つ魅力は、かなり強力らしい。

X68000
高画質

見ごろ、食べごろ!!



対応機種	PC-8801SR/TR/FR/MR/FH/MH/VA、PC-9801 SHARP X-1全機種、MZ-2500、FM-7、FM-77AV、 MSX2・X68000(黒で表示の機種は発売中です。)
メディア	5'2D5枚、5'2DD4枚、5'2HD4枚、3.5'2D5枚、3.5'2DD3-4枚
グラフィック	イメージスキャナーによる画像取り込み
メッセージデータ	20万字程度(原稿用紙500～700枚)
価格	9,800円

NOVEL WARE

THE LEGENDS OF DIMENSION ALPHA

CHATTY シャティ



ボルテージアップの為、あえて沈黙し続けたシャティ。始動!!

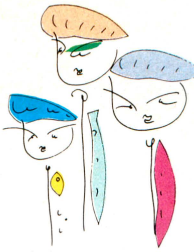
待ちわびたシャティの始動だ。今ここでシャティ情報の全ては言えないが、ヒントを与えるとすれば、ログイン1月号の「愛とは…憎しみとは…この永遠の疑問符に楔を打ちこむのはシャティかもしれない。」というコピー・フレーズだろう。また、シャティの魅力については次号から続々とニュースされるので、おいしい物でも食べてベストコンディションで待っていてくれ。

SACOM AMUSEMENT CLUB (SAC) SACは、システムサコムを応援したいという方なら、どなたでも入会できます。入会金・会費はまったく必要ありません。入会希望の方は、住所(〒)・氏名・電話番号・性別・パソコンがあれば機種名を明記の上、ハガキまたは封書で、システムサコム内「SAC事務局」までお申し込み下さい。なお、電話による申し込みは受けつけておりませんので、ご了承下さい。

ノヴェルウェアテレホンサービス……TEL.03-635-5147
ユーザーサポート……TEL.03-635-7609
電話によるお問い合わせは、月～金AM10:00～PM5:00の間、お受けしております。上記以外のお電話はご遠慮下さい。

SACOM
株式会社 システム サコム
〒130 東京都墨田区両国4-38-16
両国桜井ビル4F

資料請求券



パーソナルコンピュータとともに
EAST

FOR
SHARP X68000

日本語ワードプロセッサ　カナ漢字変換プロセッサ

絶賛発売中!!

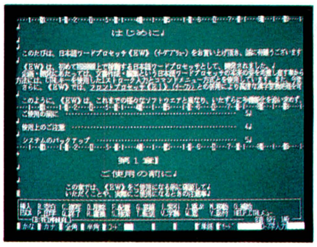
EW&E1

■個性が光るクイック・ワープロです。

SHARP X68000対応　¥38,000(E1付)

機能の数を重視する現在の日本語ワープロの中にあっては、EWは非常に個性的です。当たり前のことですが、ワープロ本来の機能と操作性を重視し、シンプルで使いやすいワープロを目指しました。ですから、スクロールなども早いですし、印刷も、わざわざメニューに戻らなくても瞬時に印刷モードに入れる使いやすさです。また、索引や目次の自動作成など、まさに文書作りに徹した個性が光ります。

■マルチウィンドウ画面



■目次と索引の自動作成



■EWの主な特長 ▶差込印刷、特定用紙印刷といった、フォームオーバーレイに対応する強力な印刷機能▶大量のドキュメント作成に非常に便利な目次、索引の自動作成▶プログラム開発等に威力を発揮するエディタモードの標準サポート▶独自のカナ漢字変換プロセッサE1の標準搭載▶他文書参照やカット＆ペーストが行なえるマルチウインド処理▶編集画面からのOSコマンド及びユーザープログラム実行▶表を含む文章での強力なブロック操作▶操作はMULTIPLANに準拠したコマンドメニュー方式▶WORD MASTERに準拠したコントロールコマンドも容易▶ファイルの大きさに制限のない仮想メモリー方式採用▶バックアップファイルを自動作成する安全設計▶OS上で稼動し標準テキストファイルを生成します。

■X68000ならではの特長 ▶イーストが独自に開発した高速日本語カナ漢字変換フロントプロセッサE1の搭載により、今までには体験できなかった日本語入力が可能です。E1は市販の代表的フロントプロセッサVJE、ATOKの良さを考慮し、設計したまさにX68000の標準となりうる高速カナ漢字変換フロントプロセッサです。▶エディタモードの標準サポートにより、行番号を意識した大規模アプリケーション開発等を行なえます。X68000の持つ優れたハードウエア機能を引き出すプログラム開発の強力な支援ツールとなります。

X68000ユーザー待望!!

パソコンが持つグラフィック機能、ミュージック機能、サウンド機能など。これらの独立したマルチ機能を統合したハイパーUD。プログラミングすることなく、絵や音が自由にエディットできるクリエイティブソフトです。パソコン紙芝居、アニメーション、パーソナルゲーム、デスクトッププレゼンテーション、各種教材、さらにビデオ編集に有効に利用できます。

GRAPHICS Editor
ペンやブラシを使って描画を
画面いっぱいにペンやブラシ、スプレーなどを使って絵や文字が自由に描けます。円や四角、直線を書いたり、塗りつぶしも思いのまま。65537色中240色を同時表示可能です。

FREE HAND Editor
マウスを使ってタイトル文字を
マウスで書いたフリーハンドの文字などを筆順に表示していきます。文字サイズ、色は、自由に変えることができます。

MUSIC Editor
メロディ、コード、リズム、パターンの設定
画面に表示された鍵盤をマウスで選択するだけの手軽さ。オリジナル曲も簡単に譜面に書き表すことができます。コード、リズム、パターンはもちろん、楽器の種類の設定もできます。

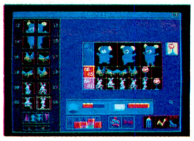
VIDEO
ホームビデオの編集もOK
スクリプトでビデオとの同期を設定しておけば、テロップ、フリーハンド、スプライト、音楽などをホームビデオと統合して使用できます。テレビ画面との統合もできます。カラーイメージユニット(別売)を使えば、オリジナルテープも。

SCRIPT



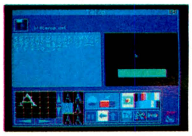
シナリオ(構成)作成も容易
各エディタで作られた映像や音の構成を設定できます。画面ごとの効果、質問文/回答文、分岐条件などを決めることができます。画面ごとのテストラン、ビデオとの同期設定やこの画面よりの各エディタの実行もできます。

SPRITE Editor



スプライトでアニメ作成を
32×48ドット、64×96ドットのスプライトが作成できます。人物や動物などのキャラクターをいくつも作成しておいて、これを続けて表示すればアニメーションやパーソナルゲームが作れます。スプライトの表示順序、速度、移動量、移動ルートが決められます。

TELOP Editor



テロップ作成も容易
あらかじめ設定しておいたテロップをシナリオの手順に従って流すことができます。文字サイズ、エッジング、バックカラーの指定は自由。テロップの方向、場所、スピードも選べます。

VOICE Editor



ナレーションの録音・再生が可能
音声デジタイズ記録ADPCMにより肉声や効果音、音楽までもファイルできます。これまでに出せなかった原音に近い自然音が表現できます。

クリエイティブソフト　カナ漢字変換プロセッサ

7月1日よりリリース

HyperUD&E1

■マルチな遊・感覚で評判のハイパーUDがE1搭載、スピードUP!ニューバージョンで新登場!!

●現在、ハイパーUDをお使いの皆様へ　バージョンアップ・サービスをいたしますので、ユーザー登録カードをお送りください。

SHARP X68000対応　予価¥21,800(E1付)

イースト株式会社　本社／〒151 東京都渋谷区代々木1-3-1 ☎(03)374-1980(代表)



Falcom
II
Ys
6月24日発売!
新発売
「イース」の優しさを継承しさらにグレードアップした「イースII」。
イースII X1版発売!
X-1turboシリーズ専用版 ¥7,800
5"-2D
4枚組
ステレオFM音源対応
ジョイスティック対応
1ドライブ使用可能
イースIIは、イースの続きです。
イースがなくてもイースIIはプレイできますが、イースをプレイしておくとイースIIがより楽しめます。
注）X-1 turbo Model-10ではディスクでDMAを使用している為、CZ-8BGR2・CZ-8BF1が必要です。
SORCERIAN
ソーサリアンX-1も好評発売中!
●X-1 turboシリーズ専用・5"-2D・5枚組・¥9,800
注）X-1 turbo Model-10ではCZ-8BGR2・CZ-8BF1が必要です。
Falcom
日本ファルコム株式会社
Personal Computer Software
〒190 東京都立川市柴崎町2-1-4 トミオービル
通信販売(送料無料)
●現金書留の場合
氏名・機種名・住所・氏名・電話番号を明記して、現金書留でお申し込みください。
●代金引換の場合
電話やFAXやハガキで、品名・機種名・住所・氏名・年齢・電話番号を明記して、お申し込みください。商品お届け時に商品代金をお支払いください。
TEL 0425(27)6501
FAX 0425(28)2714

夏のフルセール

ツクモVIPカード



ツクモVIPカード9大特典

- ファーストショッピングによる景品進呈。
- 交通傷害保険に無料加入。
- カードの盗難保険料無料。
- ご利用に応じてラブリープレゼントを進呈。
- 会員特別割引。(一部対象外)
- 全国のジャックスキャッシュディスペンサーでのキャッシングサービス。
- グッドセレクション対応。
- 「99パーソナルズ」など情報誌配布。
- 全国11万のジャックス加盟店での特別割引。

お申し込みは ツクモVIPカード事務局

03-251-9898(入会無料)

お申し込みは20才以上の方に限ります。

名古屋アメ横合同企画

グアム旅行ご招待

アメ横サマーセール

7/9～8/7

豪華なプレゼントをたくさん用意しています。1号店、2号店とも7/14～8/7は無休で営業しています。

みんな知ってる待っている 秋葉原電気まつり

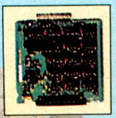
1等10万円! 6/24～8/7

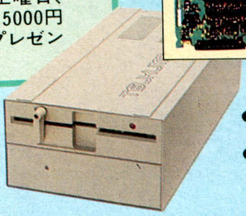
賞金総額7,000万円。

★5000円以上お買い上の方に抽選券進呈(東京店頭のみ)。

★秋葉原各店では、7月中の毎金・土曜日、PM6時～7時半のサマータイムに5000円以上お買上げの方に、サンセットプレゼントもご用意しています。

信頼と安心から生まれた ツクモオリジナル





5インチ2Dドライブ

TS-FD MKII X1

- TS-FDMKIIにケーブル及び特製I/Fをセットしたもので、これだけでディスクシステムが使用できます。
- 1ドライブはCZ-503F、2ドライブはCZ-502F相当品です。

1ドライブ特価¥32,800

2ドライブ特価¥49,800

5インチ2HDドライブ **TS-FDD MKII X1**(ターボモデル10を除く)

X1ターボ用2HD/2DD自動切替

1ドライブ特価¥38,800 2ドライブ特価¥59,800

パソコンテレビ X1G セット



- CZ-822CB……………………¥118,000
- CZ-820DB……………………¥109,800
- ディスケット(10枚)+ゲームパック……………………サービス

合計定価¥197,800

ツクモ特価販売中

モデム



オムロン	MD-1200AII	300/1200ボー	特価¥19,800
	MD-2400B	300/1200/2400ボー	特価¥39,800
アイワ	PV-A1200mkII	300/1200ボー	特価販売中
	PV-A2400	300/1200/2400ボー	特価販売中

パソコンテレビ X1 turbo Z セット



- CZ-880CB……………………¥218,000
- CZ-880DB……………………¥109,800
- ディスケット(10枚)+ゲームパック……………………サービス

合計定価¥327,800

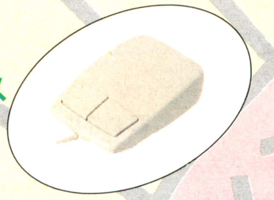
ツクモ特価¥189,800

プリンター

プリンター用紙サービス!

CZ-8PC2	カラー漢字熱転写プリンタ	特価販売中
CZ-8PK6	24ピン漢字ドットプリンタ(15インチ)	特価¥89,800
CZ-8PK7	24ピン漢字ドットプリンタ(10インチ)	特価販売中
CZ-8PK8	24ピン漢字ドットプリンタ(15インチ)	特価販売中
CZ-8PK9	24ピン漢字ドットプリンタ(10インチ)	特価販売中
IO-730	カラーイメージジェットプリンタ(15インチ)	好評発売中

X1ターボ/MZ-2500用マウス



TS-MX1 特価¥5,500

■便利なマウスパッドあります。¥1,280より

PC-E200

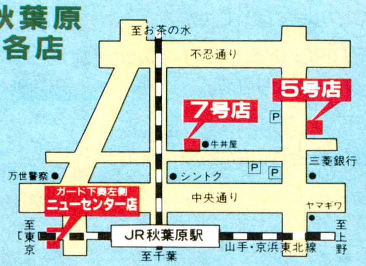
Z-80CPU、RAM容量32KB

情報処理技術者対応CASL、BASIC関数計算機能、86関数、パソコンとの接続、テキストエディタ+シリアルI/F等、機械語学習、ミニI/O機能、Z-80バス搭載。

特価¥17,800



秋葉原各店



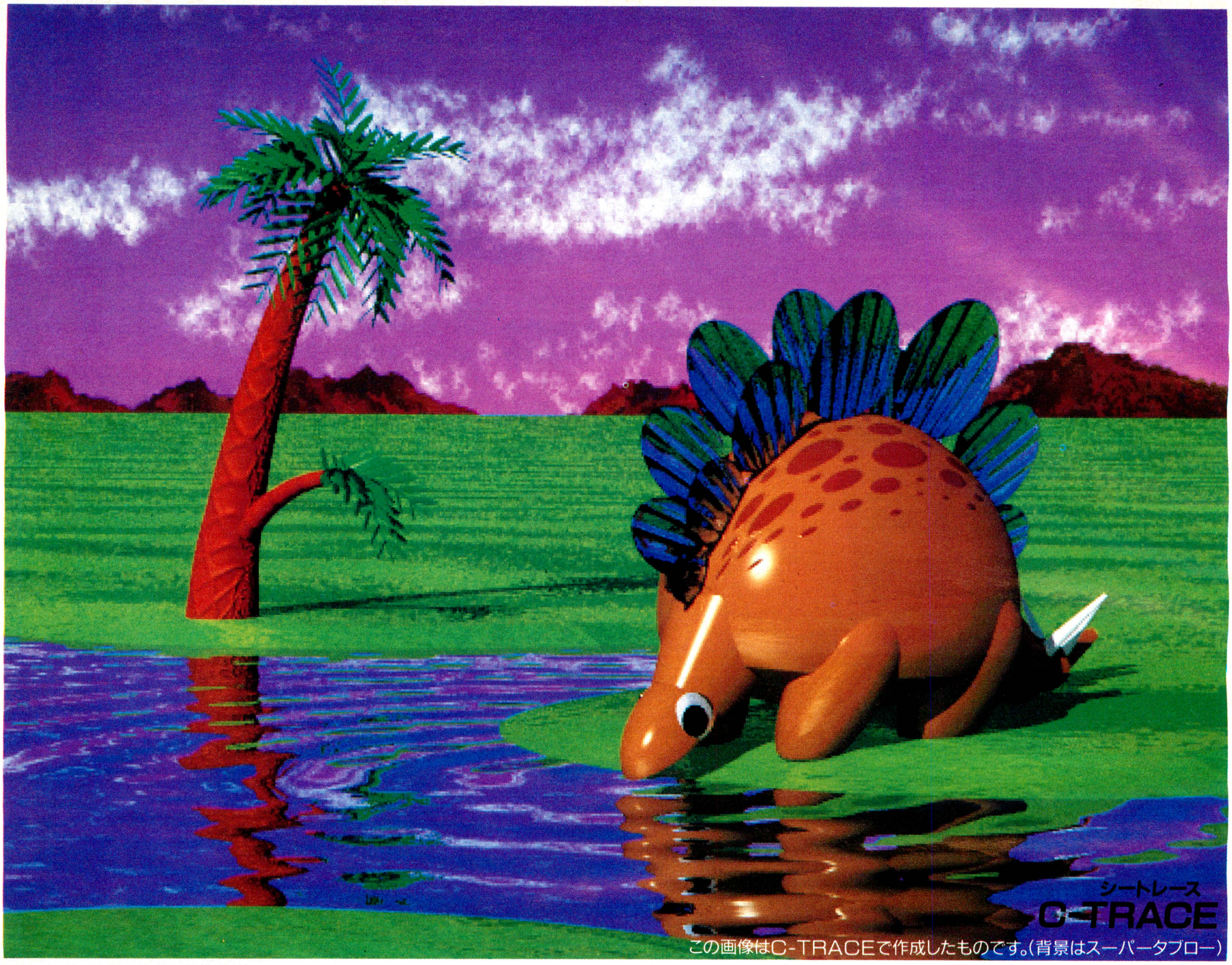
〒101-91 東京都千代田区神田郵便局私書箱135号

㊥AM10時～PM7時 ㊡6/16・23、7/14

PRO STAFF

4th. THANKS ツクモ 九十九電機

- ■秋葉原7号店 ☎03-253-4199
- ■ニューセンター店 ☎03-251-0987
- ■秋葉原5号店 ☎03-251-0531
- ■ツクモ札幌 ☎011-241-2299
- ■名古屋1号店 ☎052-263-1655
- ■名古屋2号店 ☎052-251-3399



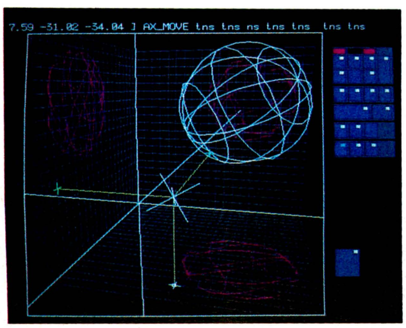
この画像はC-TRACEで作成したものです。(背景はスーパータブロー)

3次元コンピュータグラフィックス

レイトレーシングソフトウェア

	C-TRACE 98(PC-9801対応)	¥98,000
近日発売	C-TRACE 98+(プラス)(PC-9801対応)	¥198,000
	C-TRACE NEWS(SONY)	¥380,000
新発売	C-TRACE 68(X68000対応)	¥68,000

マウスでかんたん入力!



C-TRACE68は、マウスで簡単入力!
ワイヤーフレームで確認しながら形状作り。
カメラアングルもすばやく決定!
Z'S STAFF PRO-68K(ツァイト社)とのデータのやりとり可能。

C-TRACEはCF制作に活躍しています。

■動作環境
PC-9801シリーズ全機種(XAを除く)
RAM 640KB
MS-DOS Ver.2.11以上
コプロセッサー(8087,80287)有無どちらも対応

■現在サポートしているフレームバッファ(X68000は本体のみ)
PC-9801 本体内VRAM
写像 SIG社/スーパーフレーム サピエンス社
ハイパーフレーム デジタルアーツ社

■コプロセッサー,フレームバッファの販売もいたします。
10MHz 8087-1 ¥35,000

C-TRACE98ユーザーには、98+を差額(¥100,000)にて交換いたします。

Cast
株式会社 キャスト
〒150 東京都渋谷区3-6-18第4矢木ビル4F TEL.03-797-5128 FAX.03-797-6974

MICROCOMPUTER SHOW & BUSINESS SHOW

マイクロコンピュータショウ'88 &第66回ビジネスショウ

5月11日から5月14日まで東京流通センターでマイクロコンピュータショウが，また5月18日から5月21日まで東京・晴海の見本市会場で第66回ビジネスショウが開催された。今年は日程がずらされて開催されたこの2つのショウは，マイコンとビジネスという性格がますます明確となり，互いにそのコンセプトがはっきりと打ち出されたものになった。





今年は一般来場者の数も少なくなり，その専門的色合いがさらに強まったマイコンショウ。ただし，Xシリーズの参加がなくなったのはちょっと残念





AXが話題を集めた今年のビジネスショウ。シャープのブースではOS-9/X68000を始め，数多くの情報ツールが展示されていた

マイコンショウ'88

主流は国産32ビットへ

5月11日から4日間，東京流通センターにおいてマイクロコンピュータショウ'88が開催された。例年ビジネスショウと同時開催されるのだが，今年はビジネスショウより1週間先行して開催された。今回のテーマは「マイコン：かぎりなき創造の世界」ということになってはいるが，例年のことながら各ブースの展示内容はテーマとはあまり関係がないようだ。

最近の傾向としてマイコンショウではコンピュータ，エレクトロニクス関係の基礎技術が中心になっており，具体的な製品群はビジネスショウでというぐあいに分化が進んでいる。出展内容も特定用途向けLSIほか各種コントローラ，液晶パネルなどのデバイス関係が多く，以前のようなソフトハウスからの出展もついに姿を消してしまったのは少し寂しい。

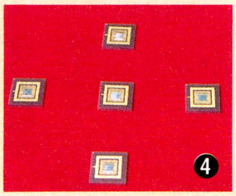
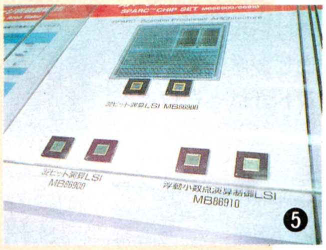
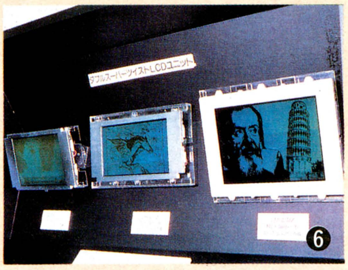
昨年はどこもかしこもASIC一色だったが今年はそれもひと段落といったところ。今回特徴的だったのが国産32ビットマイクロプロセッサに対する各社の力の入れようであった。富士通(F32)，日立(H32)などの「TRONチップとしても使える」というもののほか，VMテクノロジー（VM8600S)，NEC(V70)，そのほか最近流行のRISCチップ，Sun-4で採用されて話題となったSPARCプロセスによるプロセッサなどが展示されていた。面白いのがVM8600Sというチップで，これは従来マイクロコードで行われていた命令制御をプログラム可能な素子群に置き換えて8086のコードなどを実行できるようにしたもの。いわゆるファームウェアを実現したチップである。メーカーでは「新世代32ビットバーチャルマイクロプロセッサ」と呼んでいる。

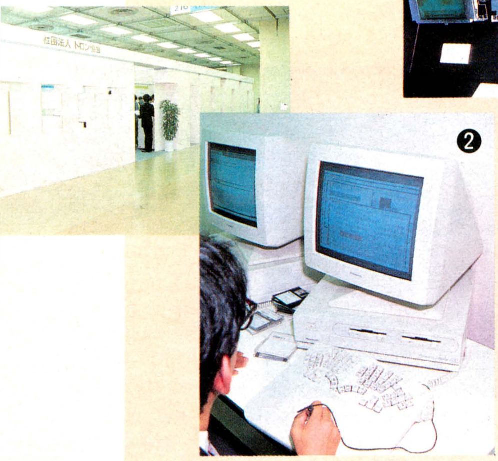
シャープのブースではXシリーズ関係の出展はなく，MZ-2861によるイメージ処理やパソコン通信関係の展示および，可変長PCM録音用LSIや光磁気ディスク，液晶/プラズマディスプレイなどの展示物が中心だ。なかでも液晶のトップメーカーとしての実力を示す，ハイコントラストなダブルツイスト液晶による表示デバイスなど，新技術によるディスプレイパネルが参考出品されており注目を集めていた。今後のポータブ

マイコンショウ

❶MZ-2500用時計システム。日本マイコンクラブの出品
❷トロン協会はブースもTRON。今年は動いていたTRONマシン
❸シャープと三菱による非ノイマン型プロセッサ
❹VM8600Sは32ビット仮想プロセッサ
❺富士通のSPARCチップ
❻ハイコントラストな新型液晶ユニット
❼パソコンの画像をOHPに出力（参考出品）
❽MZ-2861によるイメージ処理システム









❾AXマシン勢揃い。上からシャープAX386，三菱MAXY・M3201-A20/M12，三洋MBC-17Jシリーズ

ル機器の表示デバイスを占ううえでも興味深い技術だといえるだろう。　(S. N.)

第66回ビジネスショウ

AXマシン登場

東京・晴海の見本市会場で開催された第66回ビジネスショウは，パーソナルコンピューターユーザーをワクワクさせてくれるような新製品が登場しなかったこともあって，今年の会場全体のイメージとしては，比較的こぢんまりとまとまった印象のものであった。特にここ数年のAIや通信，はたまた電子黒板といった，会場内のあちこちで見かけられるような流行の中核となるものもなく，その分自社の目玉商品を前面に押し出し個性的な演出を心がけたブースが多い。

そのような雰囲気の会場のなかで，今年のテーマが「インテリジェントオフィス・仕事とやすらぎの展開」だったせいもあってか，事務機メーカーの各ブースは演出にも凝っていて，赤と黒の大胆なカラーを使った屋根付きブースを用意したり，派手なデモンストレーションをしたりと，今回のショウのなかではいちばん活気があった。また，コンピュータ関連のブースのなかでは，MC68030搭載の新製品・NWS-1830/1850を発表したソニーが，ブース全体をNEWSシリーズのハード/アプリケーションで統一した，地味ながらもポイントを絞ったデモンストレーションを展開しており，多くの来場者の関心を集めていた。

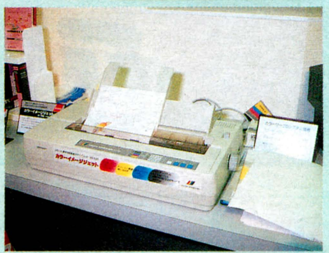
今年のビジネスショウ全体のなかで話題を集めたものといえば，やはりシャープ，三菱電機，三洋電機の3社が投入してきた日本語PC/AT互換のAXマシンということになるのだろうか。三洋電機のMBC-17Jシリーズは事前に発表されており，マイコンショウにも登場していたので，さほどのインパクトはなかったものの，ビジネスショウには三洋がラップトップも含めた11機種を同時に発表し，特設コーナーでの国際色豊かなデモンストレーションには驚かされた。そのほかシャープ，三菱電機もブースの前面にAXコーナーを設置していたが，それらのデモがいずれもMS-WindowsやGuide，IBM-PC用ゲームのほかに，一太郎などの日本の既存ソフトの移植版を用意したところも多く，来場者もこのハードに関して注目はしているものの，期待度の点











⑩豊富に揃えられたパーソナル情報ツール。上からノートワープロWV-500, カラーワープロWD-910, 参考出品の電子ダイアラー, ポケコンPC-E200/E500
⑪新しく登場したシャープのプリンタ3タイプ。レーザプリンタMZ-1P23, カラーインクジェットプリンタIO-730, 136桁漢字プリンタMZ-1P30
⑫これがマルチタスクで話題のOS-9/X68000
⑬真剣な表情で原稿を読み上げてはワープロに文字を入力/変換させていた, 音声ワープロのデモンストレーション
⑭ズラッと並んだソニーのNEWSシリーズ
⑮参考出品されていたPC-9801版Kamikaze













からするといまひとつといった印象だ。

トータル情報システムをテーマに

さて，期待しながら訪れたシャープのブースでは，電子手帳からワークステーションまでと幅広くハードや各種ツール類が展示され，「知的活動を高度に支援するトータル情報システム・キャンパスOA」のテーマどおり，個人ベースから企業規模までに対応できるビジネスツールがところ狭しと展示され，トータル情報システムを前面に打ち出したシャープブースに対する一般来場者の関心も高かったように思う。

そのなかでも最大の関心はやはり初登場のOS-9/X68000。このOS-9のマルチタスク・マルチウィンドウのデモには X68000ユーザーのみならず，他機種ユーザーや各企業の開発担当者から熱い視線が送られていた。OS-9/X68000の詳しいことについては今月の「X68000あなたの知らない世界」(116ページ) に紹介してあるので，そちらを参考にしていただきたい。

また，今回シャープが発表したAXマシン・AX386-F/386-FH4の2機種は，ブースの前面に大きくスペースをとって展示されていたが，シャープのAXマシンは32ビットのみで価格も80万円以上と，X68000と同じくこのAXマシンもパーソナルワークステーションと呼ばれているようだが，この2つではかなりニュアンスが異なっているのを実感した。

そのほかには光ファイルシステムDQ-5000やΣワークステーション，カラーワープロWD-910，ノートワープロWV-500， 光磁気ディスクドライブJY-500，電子手帳PA-7000/6500，ポケコンPC-E200/E500などなど，参考出品のパーソナル音声ワープロまで含めると，その内容の豊富さには目を見張るものがある。個人的には薄型ノートワープロWV-500の軽量小型化に興味を引かれたが，いずれシャープのラップトップ型パソコンが登場するのであれば，ぜひこのサイズに挑戦してほしいものだ。

今回のショウのように，シャープがトータル情報システムを提案するのは企業カラーから考えても大いに賛同したいところだが，今後はこれら既存のハードとこれから登場するであろうさまざまなハード/ソフトとが，フルコンパチブルなトータル情報システムを完成してくれるその可能性に期待しておきたい。 (T.S.)

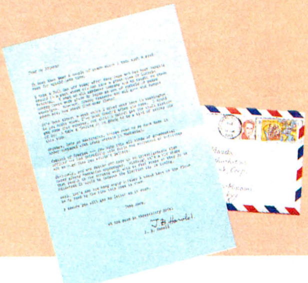
THE SOFTOUCH

SOFTWARE INFORMATION





うわーい，X68000でカーレースが遊べるぞ，おまけにドッジボールまでできちゃうぞ。おまけに，J.B.ハロルドから手紙までもらったぞっと。





話題のソフトウェア

どうです，いきなり登場した上の写真を見て驚いた方も多いことでしょう。ここしばらく，やれ「ドラスピだ」とか「R-TYPEはまだか」とか，シューティングゲームの話題ばかりが先行していたX68000のゲームソフトに，いきなり**フルスロットル**と**熱血高校ドッジボール部**ですからねえ。でも，シューティングゲーマーとしての血が騒ぐ，なんぞといいつつ，首をながーくしている皆さんにも，この2本のソフト登場は一服の清涼剤として十分楽しんでいただけるのではないかな。この2本はいずれも**7月中**にシャープさんから発売されるので，期待していてね。

そうそう，5月に突然Oh！X編集室にエアメイルが届きました。差出人はというとあの**J.B.ハロルド**。いま桜の季節も終わったワシントンでバーボン片手に難事件に取り組んでいるらしい。さて，この結末をX68000上で報告してくれるのはいつになるのかな。それからそれから，膨大なスケールで話題を集めているシミュレーションゲーム**太平洋の嵐**も夏か秋ごろ登場するんだって。これまで「シミュレーションゲームが，X68000には出ないじゃないの」といっていた方，大丈夫です。きっとこれ1本で，来年のお正月までずっと遊んでいられるはずですから。

さて，来月のSPECIAL REVIEWは**ソーサリアン続編**と**イースII**の豪華2本立ての予定です。7月にはHuCARDに**ファンタジーゾーン**も発売されるようだから，こうなればまた近いうちに，ゲーム特集でワッと盛り上がってみたい気分になってきましたね。



読者が選ぶ今月のゲームベスト10

おのれ，世の平和を乱すにっくき頼朝め，地獄から復活したこの景清が，みごと成敗してくれるわ！　というわけでダントツトップの源平討魔伝です。X68000版が発売されて以来，うなぎのぼりの大人気。満点のサウンド効果，画面狭しと出没する妖怪変化，だじゃれの国に血の池地獄，ひえぇ，ハマッちゃったよー，という人が続出してるらしい。一方，前回トップのM＆Mやイースなどのファンタジーも変わらず強力です。ソーサリアンも出てきたし，「剣と魔法」の世界はこれからますます目が離せなくなってきましたね。

1．源平討魔伝
2．スーパーレイドック
3．Might and Magic
4．イース
5．上海
6．SUPER大戦略
7．三国志
8．ぎゅわんぶらあ自己中心派2
9．スペースハリアー
10．ウィザードリィ

新作ソフト情報

☆…6月4日現在発売中　　★…近日発売予定

★ハイドライド3

やったー，今度はハイドライド3がついに登場するぞ。舞台はもうすっかりお馴染みになった人間と妖精たちが平和に暮らす国，フェアリーランド。その国では過去何度かの苦難の時代があったが，いまは宮殿に祭られた3つの不思議な宝石の力によって，平和が保たれていた。しかし，ある日，これら3つの宝石が何者かの手によって持ち去られてしまったのだ。すると過去，人々をいまわしい目に遭わせたあの悪魔バラリスが復活し，再び王国全土を恐怖のどん底に陥れようと動き始めた。そうして美しいアン王女にまでバラリスの魔手は迫った。いざ，王国とアン王女を救うため勇者は立ち上がるのだ。ここから新たなハイドライド伝説が始まろうとしている……。

X1/X1turbo用　　5"2D版2枚組　7,800円
（2ドライブ専用）
ティーアンドイーソフト　　☎052(773)7770

★イースII

ソーサリアンが出たっ！　と，大騒ぎしている間もなく，今度はこのイースIIが6月24日に発売される。ストーリーはイースIで集めた6冊の古文書の謎を解き明かすため，再びアドルは氷や溶岩世界そしてサルモンの神殿へと向かうことになる。マップは前作の4～5倍，数多くの謎を含み前回以上に力のこもった24曲のゲームミュージックとともに，感動のラストシーンへと，プレイヤーに息をもつかさぬシーンが次々と展開する。ソーサリアンのパッケージに書かれていた「X1版はデキがいいよ」の文字が，またこの「イースII」でも実現されること大いに期待しておきたい。

X1turbo用　　5"2D版4枚組　7,800円
（Model10は要CZ-8BGR2, CZ-8BF1）
日本ファルコム　　☎0425(27)6501

★めぞん一刻・完結編～さよなら，そして……

明るい一刻館の住人たちが繰り広げるAVG,「めぞん一刻」が帰ってきた。今回は一刻館に八神が転がり込んで，またまた五代と響子，そしてお馴染みの住人たちを交えて大騒動が巻き起こる。果たして五代は本人に向かって「響子さん好きじゃー」コールができるのだろうか。そして響子さんの花嫁姿を最後には見られるのだろうか。

X1/X1turbo用　　5"2D版2枚組　7,800円
マイクロキャビン　　☎0593(51)6482

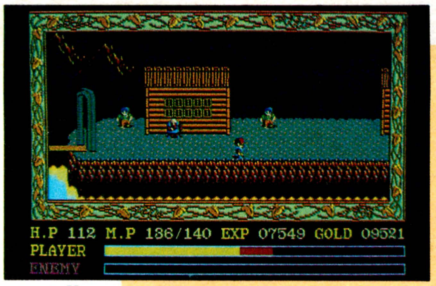
★第4のユニット2

4月号のゲーム特集で，ユニークなゲーム構成で好評だったあの「第4のユニット」が，さらにバージョンアップしての登場だ。メッセージは前作の2倍，パワーアップされた戦闘モード。新たな強敵ダルジイの登場によって，ハードアクション・ハイパーバトルの世界が，いま再び始動する。

X1/X1turbo用　　5"2D版3枚組　7,600円
データウエスト　　☎06(968)1236

★ソリティア・ロイヤル

トランプゲームのピラミッド，ゴルフ，クロンダイク，シャッフル＆ドロー，神経衰弱など11種類のカードゲームを集めた「ソリティア・ロイヤル」が発売される。カードゲームは遊びの基本，さらにそのなかから，アメリカで人気のあるカードゲームを一挙に集め，それらの1つひとつのゲームで高得点を目指すプレイモードと，すべてのジャンルのトータルで勝敗を競うツアーモードの2つのモードがあり，それぞれ得点がセーブされハイスコア機能も付いている。また，初心者は練習モードで腕を磨いて本番に臨め，おまけにマウス対応の親切設計となっている。



イースII

X1/X1turbo用　　5"2D版　6,800円
ゲームアーツ　　☎03(984)1136

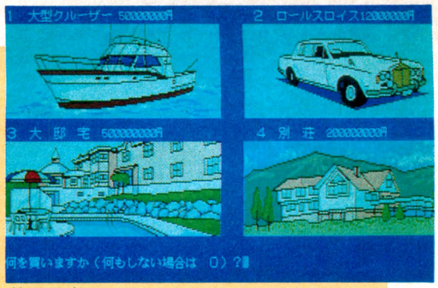
☆SUPER DEVICE MONITOR"T"

このSUPER DEVICE MONITOR"T"は，これまでパソコン通信でアスキーコードに一度変換してからオブジェクトデータを転送していたのに対し，このツールを使えば直接オブジェクトデータのまま特殊な圧縮方法を用いてデータを転送し，転送速度を従来の最高32倍までアップできる。このほかに姉妹品としてX1/X1turbo共用の「SUPER DEVICE MONITOR"T"」も発売されている。

X1turbo用　　5"2D版　13,000円
BLUE SKY　　☎0559(72)6710

★億万長者

ここ数年の地価の上昇で「億万長者」なる言葉にもあまり有難みがなくなってきた今日このごろ。しかし，いつの時代も大金持ちになる夢は捨て難いもので，せめてゲームのなかの世界だけでも億万長者になる夢とチャンスを与えてくれるというのがこのソフトなのだ。要は各国の貨幣価値をにらみながら，株や商品相場などに手を出して持ち金を増やしつつ大邸宅やロールスロイスを持てる生活を追い求めるというもの。たとえゲームのなかだけとはいえ，大金持ちになったときの気分というのは，やはりなんともいえない快感だったりするのである。

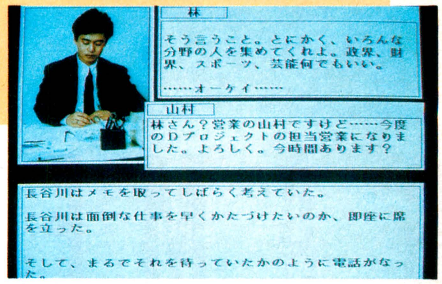


億万長者

X68000用　　5"2HD版　9,800円
コスモス・コンピューター　　☎03(770)1821

☆ドーム

システムサコムがすでにX1版で発表しているノベルウェアシリーズの「ドーム」がX68000に移植された。内容は，都市をガラス状のドームで覆って放射能汚染から人類を守ろうとスタートされたプロジェクト「ザ・ドーム計画」。そのドーム計画に絡んで闇にうごめく数多くの人間たち。果たしてこのドームに隠されている真実とは，いったいどのようなものなのだろうか……。画像取り込みを多用したグラフックやマルチウィンドウ，そして文字表示スピード設定機能など，X68000ならではの工夫が随所に見受けられる新作ソフトだ。



ドーム

X68000用　　5"2HD版5枚組　9,800円
システムサコム　　☎03(635)7609

"まにあ"の方必見のゲーム本が出た!!

「まにあ」の人が待ちに待ってた(らしいです。わたしはマニアじゃなかったから知らなかったけど……)『テレビゲーム電視遊戯大全』がUPUから出ました！　うーん，もうこれは「スンゴイデスネー」と所さんふうに驚くしかない本ですよ！　だって，いきなり本のページが2つにも3つにもボコボコ切れてたり(どういうふうになってるかは実際に買ってみてのお楽しみね！)，油断してると本がバインダーになっているのに気づかずにページがばらけてしまったりと，見かけが油断のならない格好をしているうえに，内容が地獄の修羅界です。

そう，アメリカ，日本，果てはイギリスやフランスまで含めた世界中のゲームに「どっぷり」なマニアック本なのです。ポン（これって世界最初のTVゲームなんですよ！　知ってた？）からR-TYPEまでのゲーム家系図（本当の名はゲームマップという），古今東西のゲーム名薄（終電の電車のなかでこれを友だちと見ると「あ，ギャラガだ懐かしいなー」，「ローグまであるぅ！」と騒音地獄と化す），揚げ句の果てには遠藤雅伸さんからボールブレイザーのデビッドフォックスさんまで世界中のゲーム作家にインタビューしまくってしまうものすごいパワーです。そのパワーはこれからゲームを作ってみたいと思う人に，非常にはっきりとゲームの思想を見せてくれます。ゲームが好きな人なら35回ゲームを我慢しても買う価値のある必見の1冊だと思います。　（で）

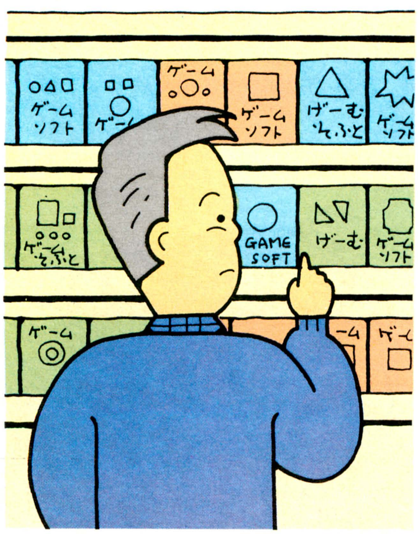


テレビゲーム電子遊戯大全 3,500円
UPU　☎03(235)7561

THE SOFTOUCH

GAME REVIEW

今月はX68000に魔神宮の続編「グランド・マスター」を，そしてX1にはちょっとタイプの違った将棋練習用ソフトと野球ゲームをご用意しました。いずれも派手さはないけれど，工夫の見られるゲームですので，ごゆっくりお楽しみください。



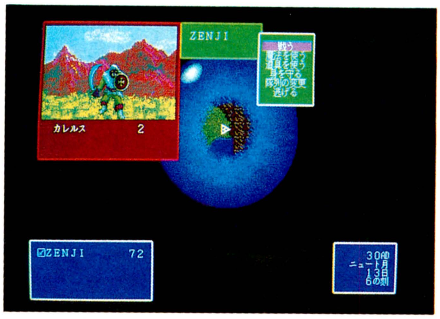
グランド・マスター

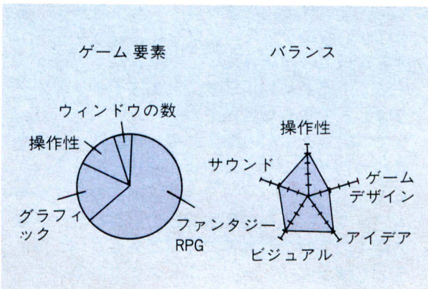
大陸グランツで繰り広げられるファンタジーRPGの世界。中央に開いた丸いウィンドウが斬新だ。

▶宝珠の向こうに見えるものは，大陸グランツだった。このグランド・マスターは，X68000の得意技であるウィンドウをバシバシ使って，小気味よいゲームである。本格的RPGが少なくなっているなかで，これは掘り出しものソフトといえる。店のおっちゃんにはスパイを頼まれるわ，おばはんには手紙を渡されるわで，私は信頼されているのだなあと実感できる仕組みである。しかし，十分なお礼をするからという甘い言葉を信じて引き受けても，渡されるのは雀の涙ほどの金額である。このようにこのゲームは登場キャラとの会話も楽しめる。獅子が子を谷底へ突き落とすように，バンバン戦って生き残った者だけを集めて最強メンバーを組むのだ。そうすれば，マップが見にくいだとか，敵は貧乏だとか，魔法は役立たずだとかいう文句は聞こえなくなり，余裕で戦いを満喫できるだろう。X68000で本格的RPGを楽しみたい人にはお勧めの1本といえる。

熱中度 ▶▶▶▶▷▷▷ (A.N.)

▶極めてアリガチなタイプのRPGである。ゲーム自体はそれほど難しいわけではない。どちらかというと謎解きがメインで，敵はそれほど出てこないからだ。行く先々でいろいろなことを頼まれるというソーサリアンタイプのゲームなのである。







つまるところ最近の流行をミックスしてできた，ごく一般的なRPGといったところであるから，RPGをやってみたいという人や，長く楽しみたいという人にはいいかもしれない。

しかしこのゲームにはかなり不満がある。あまりにもX68000の性能に甘えているのだ。X68000をもってすればこのぐらいのゲームは少し腕の立つ人だったら簡単に作れる。X68000なんだから画面構成にしろ操作性にしろもっとマトモにできるはずなのに。もう少し努力してほしいものだ。

それにマニュアルも最低である。もっと丁寧に書いてくれないと，なにがなんだかわからないじゃないか。プンプン。

熱中度 ▶▶▶▷▷▷▷ (C.W.)

X68000用　5"2HD版 2 枚組 9,800円
ザイン・ソフト　☎0794(31)7453

振飛車

これは実際にプレイするのではなく，問題を解きながら，または対局譜を見ながら実力アップを図る練習用将棋ソフトなのだ。

▶え？　これってゲームレビューでしょ。このソフト，ゲームというよりは「教育ソフト」じゃないかと思うんですけど……。なんとこのソフトはあらかじめシナリオ(プロの指した棋譜だそうな）が決まっていて，ポイントとなる場面で4択式の問題が出され選んだ手によって点数が加算され，プレイヤーが何級であるかを認定してくれる「将棋版オート家庭教師」というとんでもないソフトなのです。そこで，私も実際に問題を解いてみたのですが……。エーン，なんなんだよこれはーっ!!　問題のヒントが「厚く指したい」とか，「5二飛に対応する動きは」とか，なんだかよくわからないヒントばかりのくせに（単に私が無知なだけだが)，人がチョット指し違えたぐらいで「悪手」とか，「不満が残る」とかいってすぐ0点付けるんだもんなー。ヘタで悪かったね，

ヘタで！　どうせ指導するなら，もっと親切に指導してよねー。ぶつぶつ……。

熱中度　▶▶▶▷▷▷▷　　（で）

▶わたしゃ，いままで将棋ソフトなんてもん買ったことがありません。まぁ，単純に将棋が下手だから，という理由ですけどね。そんな私が一目置いたこの将棋ソフトは，コンピュータを相手に勝負するものではありません。

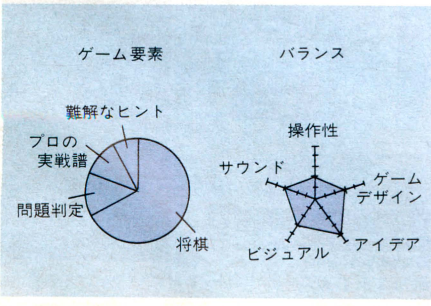
こいつは，プロが指した手を画面上に展開していって，何手か指したときに「ここで問題です。あなたなら次の一手，どう指しますか？」なんていうスポーツ新聞の片隅でやってるようなことを，パソコンでやってくれるのです。収められている問題は初級から上級まで全部で30局あって，さらに，一局につき10個問題が用意されています。解答の方法も番号を選ぶだけで，至って簡単なんですよね。おまけに，棋力認定ソフトなんて肩書きも持っていて，段・級位を教えてくれます。まさに，至れり尽くせりのソフトなんです。でも，でもね，操作性の悪さがこのソフトの全体のイメージを落としています。それでも，将棋好きのマイホームパパなら満足するかな？

熱中度　▶▶▶▷▷▷▷　　(H.K.)

X1turbo用　5″2D版2枚組 7,800円
ビクター音楽産業　☎03(423)7901







Mr.プロ野球

常にリーグ優勝を目指しながらも，他球団との駆け引きや球団運営まで，幅広い監督としての采配がものをいう野球ゲームだ。

▶先日，東京ドームへ日ハム―南海戦を見に行った。南海が負けたので悲しかった。誰かが東京ドームの野球は“箱庭野球”だといっていたが，まったくそのとおりだ。きれいで風もないし，寒くないのはいいかもしれないが，失ったもののほうが多いと私は思う。どうせ箱庭野球なら「Mr.プロ野球」のほうがときどき手を下せるだけ面白い。たまに試合に目をやって，危ないと思ったら投手をブルペンへ派遣し交代させればいい。TVを見ていて「あっ，俺だったらこうするのに」を実際にできるのだ。試合終了後，なけなしの資金でチーム強化策を考える。オーダーとローテーションで悩み，試合が始まったら適当に観戦しながら手を下せばいいから，究極のヒマつぶしソフトといえよう。私は中日を選び，巨人，西武，南海，大洋でリーグを作ったが，弱そうなチームばかりを相手にするのもよい。昨日，落合が故障して全治6日と診断された。戦力低下が心配だ。気合いモードが欲しい。

熱中度　▶▶▶▶▷▷▷　　(K)

▶球場が横向きになった風変りな野球ゲームである。ボールも投げなきゃ，バットも振らない。やることは選手の交代，盗塁の指示など，つまり監督さんである。勝っていればいいが，負けてくると「この馬鹿」などと罵り，「俺に打たせろ！」と騒ぎ，「明日は特訓だ」と堅く心に誓うのだった。

試合が終わると，なぜかいきなりオーナーになって球団の雑務をする。球場の整備や，選手の訓練，果ては写真週刊誌対策までしなければならない。このあたりは，いわゆる経営シミュレーションゲームになっているのだが，あれもこれもと欲張ってピンからキリまでいろいろな要素を混ぜたために，全体的には散漫になっているような気もする。

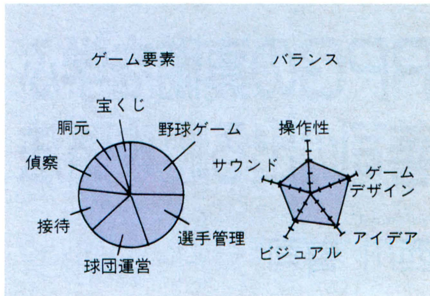
これを1年分繰り返して，最後に優勝していなければならない。あまり優勝できないとクビになる。この本が出るころも巨人は負け続けているのだろうか。

熱中度　▶▶▶▷▷▷▷　　(M.Y.)

X1/X1turbo用　5″2D版2枚組 7,500円
（要漢ROM，2ドライブ専用）
ブラザー工業　☎052(263)5895







時代はいつも諸行無常なのである

何年たっても九段下駅前の地下工事は終わらない。しかし，ほかでは時は流れて風は吹くのである。ゲームだって昔よりきれいな絵でありたいし，たくさんデータが欲しいと思うのである。メモリだってたくさん食べたい。最近X1turbo専用のゲームが多いとか，さらには2ドライブ専用が多いなどとお嘆きの貴兄が多いが，ゲーム屋さんだっていろいろな注文に対応するのはたいへんなのである。

確かに，X1でも作れそうなものをX1turbo専用にしてしまうのは，ソフトハウスの怠慢以外のなにものでもない。だが，このところの事情は変わりつつある。88なんかとうの昔にSR以降の機種に絞られたし，AVより前の7/77も切り捨てられてしまったではないか。しかもよく考えてみると，X1よりもX1turboのほうが機能が高いわけだから（値段も高いし），同じソフトしか走らないとしたらX1turboユーザーは損である。元MZ-2500ユーザーの私から見れば，5年以上も前のハードで最新ソフトを出せというのはわがままだと思う。そりゃX1twinユーザーには悪い気はするけど。　(K)

THE SOFTOUCH

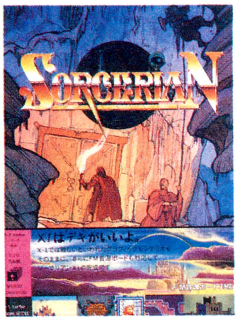
●ソーサリアン(その1)



RPGの真髄を学ぶ善司の出る順攻略法

Nishikawa Zenji
西川 善司

待望のソーサリアンがついに我々の前に姿を現した。このゲームはぜひともみんなが納得するまでレポートしてあげたい。というわけでM&Mに引き続き，「完結するまで何カ月でもご紹介シリーズ」がここに再び復活するのです。



X1turbo用　5"2D版5枚組 9,800円
(model10では要CZ-8BGR2，CZ-8BF1)
日本ファルコム　☎0425(27)6501

それではスタートのご挨拶

このゲームには特にそれらしいストーリーはありません。全体を流れるストーリーというより，“ペンタウァ”というソーサリアンの町にかかわる数々の試練（冒険）をキャラクターが乗り越えていくことにより，自分だけの「ソーサリアン」の世界を作っていくというゲームなんです。普通のRPGのように，ひとつの（ゲームの作者がお膳立てした）ストーリーをただ追いかけるのではないのです。

ですから，プレイヤーがキャラクターを生み出したときから，彼らの生活が始まるわけです（ファイターという冒険者としての役割を持ちながらも，冒険に行かないときには町で鍛冶屋をやっていたりする）。こういうRPGの観念には，正直いって私は驚きました。このソフトの9,800円は安くはないけれど，決して高くはありません。

キャラクター作りが重要

さあ，ノリのいいBGMにのってキャラクター作りといきましょう。作れるキャラクターの種族は4種類，ファイター，ドワーフ，エルフ，ウィザードです。作れるキャラクターは10人まで。ちょっとここでアドバイス。もちろん10人全部作るのも結構。ただし，ソーサリアンは冒険に出て帰ってくると1年が過ぎてしまいます（冒険に出たキャラが1歳年をとるのはもちろん，町に残ったキャラクターたちも年をとる)。シナリオは15個ありますから，これらを全部クリアするのには最低ゲーム上で15年かかるということです。

生まれたてのキャラクターは，最年少でもすでに16歳ですから，15のシナリオをすべてクリアすると，最高で16＋15＝31歳になってしまいます（まず，こんな若さで終わることはあり得ないが)。

すると，今日はこのキャラクターで冒険しよう，今日はこっちのキャラだ，とやっていると全体的にオジンキャラの集団になってしまいます（この場合，ちゃんとジイさん，バアさんのグラフィックになるところがまた芸が細かい)。

それに，ファイターやウィザードはほっといても60歳くらいで寿命をまっとうしてしまいますから，なるべく特定のキャラで遊ぶのがいいでしょう。一度エンディングを見たあと，ゆっくりと「ソーサリアン」の世界に浸るというのも悪くありません。

話をキャラクター作りに戻しましょう。マニュアルを読んでのとおり，INT（インテリジェンス，知力）が0以下だと魔法が使えません（ソーサリアンは顔じゃなかった魔法が命です)。ファイター，ドワーフはINTがマイナスですから，魔法は使えないと思われますが，実は修行などをしてINTをプラスにすれば使えるようになります(マニュアル参照のこと)。

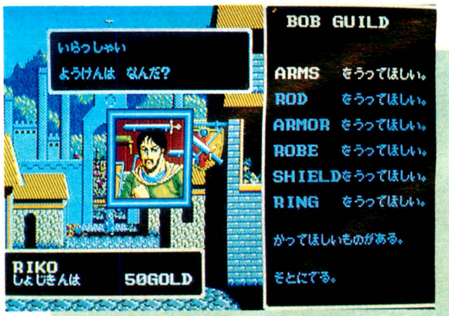
ただし，修行して無理やりプラスにしてもファイターなどは，レベルが上がるたびにINTが減ってきてしまいますからいつかはマイナスになってしまいます。これを，なんとか防ぐ方法はキャラクター制作時にBONUS値をINTに多めに割り振ってやるのです（もちろんBONUS値だけではプラスにならないかもしれないけど)。

要するに，キャラ制作時におけるBONUS値の割り振りは今後のキャラクターのレベルアップ時のパラメータの変化に対して加速度的な役割をするということなのです。ちなみに，私のドワーフの男はキャラ制作時にINTにBONUS値を与えすぎたので，魔法ONLYの剣の振れない奴になってしまった（ドワーフなのにSTRがマイナスというのが怖い)。決してひとつのパラメータばかりに値を集中させないように！

正しいキャラクターの作り方

ソーサリアンでは，RPGお決まりのパターンのファイター，ウィザードなどの種族を決定したほかに職業を60種のなかから選べます。この職業というのは，冒険をしないキャラクターが（冒険に出かけたキャラクターが帰って来るまで)，町でなにをやって過ごしているかというものです。1年間同じ仕事についていると，ある程度の経験値が貰え，キャラクターのパラメータ(STR, INTなど）が変化します。つまり職業とパラメータは密接な関係にあるのです。私は鍛冶屋がなかなか美味しいと思うんですが（ちなみにデフォルトは農夫です)。

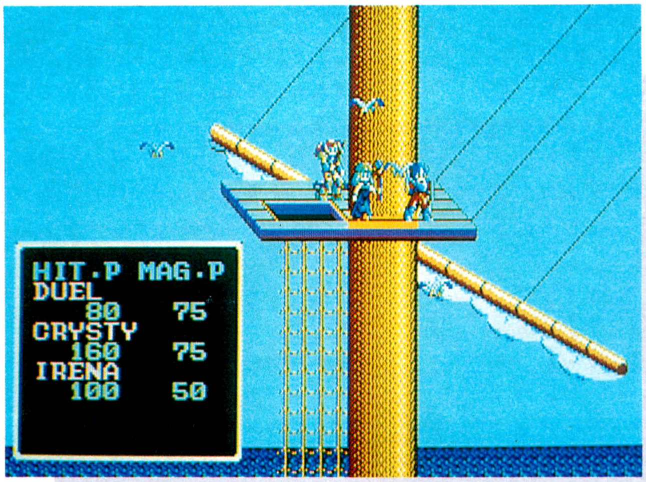
キャラクターのパラメータは，ゲームにとても大きな影響を与えます。たとえば，高レベルのシナリオになればなるほど，扉



最初はまず50ゴールド持ってお買い物

が重くなって，VIT（活力，腕力）の値が小さいと開きません（ちなみに「呪われたクイーンマリー号」ではVITが27〜28以上ないと扉が開かない)。また，レベルの高いシナリオでは，敵の攻撃力が上がっていますからPRT（防御力）も高くないと，死んでしまうかもしれません。パラメータのバラつきが見え始めたら速やかに町の道場へ行きましょう。

えっ？ 修業は2年もかかるからイヤだって？ 馬鹿なこといっちゃあいけません。88版なんか3年かかるんですからねっ（おかげで私が先に終わらせた88版のキャラたちは全シナリオクリア時には全員ジイちゃん，バアちゃんになってしまった)。もっともKRM（カルマ）の値を上げれば1年で修業を終えさせてくれますけど。さらに，道場では特殊な知識を修得することができます。得られる知識はアイテム，罠，モンスター，ハーブの4つですが，私が思うに，実際にないと困るのはアイテムと罠の知識くらいでしょう。







左上から順番に「呪われたクイーンマリー号」，「失われたタリスマン」，そして「ロマンシア」です。ソーサリアンでは，このようにそれぞれのシナリオによって舞台となる場所が工夫されていて，楽しさ倍増の大サービス

アイテムと魔法の話

あの名作と呼ばれたウィザードリィやファンタジアンなどのように，アイテムはたくさん集められますが，ソーサリアンでは絶対に

魔法＞アイテム

なんです。そう，ソーサリアンは魔法が命です。人形は顔が命です。

ソーサリアンにおいては，装備することのできるアイテムにはすべて魔法がかかっています。しかし，その魔法はすべて町で売っているアイテムにも，お金を払えばかけることもできるんです。ですから冒険中で得られたアイテムは，売ってしまっていいと私は思います。おっと，アキカンが飛んできた。えっ？ あぁそうだ，忘れてた。冒険中に得たアイテムは売る前に必ずアイテムの知識を持っているキャラクターで一度鑑定してから売るようにしましょう。なぜかって？ じゃあ，その理由をお話ししましょう。

ソーサリアンの魔法は火星，水星，木星，月，太陽，金星，土星の7つの魔法の要素の組み合わせによって魔法がかかることは，マニュアルを読んですでにご存じのことかと思います。そして，火星はSTR(攻撃力)，水星はINT，木星は……（以下マニュアル参照のこと)，などのパラメータを上げる効果があります。火星が5個かかっているとするとSTRが5つ増えるわけです。ですから魔法の使えないキャラクターでもパラメータを上げる手段としてアイテムに魔法をかけるのもいいでしょう（修行をするよりこっちのほうが早いし)。

たとえば，SUN RAYという魔法は，火星，木星，太陽，金星がアイテム（もちろんどんなアイテムでもいい）に宿ると使えるようになるのですが，冒険で得られるガラディーンという剣にもSUN RAYの魔法がかかっています。しかし，鑑定してみれば判ると思いますが，ガラディーンには，な，なんと火星が10個，太陽が6個かかっているのです！ ここでなにを興奮しているかというと，町では同じ魔法の要素は5個までしかかけられませんから，これはラッキーというわけです。火星10個といえば攻撃力が10も上がるんですよ。こういうアイテムを売ってしまっては大損ですね。というわけですから，アイテムは売る前に鑑定しましょう。

魔法の話が出てきたところで，耳よりな情報をひとつだけここで紹介しておきましょうか（表1も見てね)。これまで読んでいただいた文章には，実は誤りがひとつ含まれています。それはどこかというと，「魔法をかけてもらうのは修行より早い」なんて書いてあったところです。おぉ，善司はなんて「タワケ」なんだと思った方もおられたでしょう。だってマニュアルに「修行は2年かかり，魔法はひとつの要素をかけ終わるのに3年かかる」とありますものね。

それでは，試しに武器屋で適当なものを買ってきて，魔法屋に行ってみてください。そして，適当な魔法をひとつかけてもらってください。すると「3年かかりますけどいいですか？」と聞いてきますね。ここで当然「OK」を選択します。そして次に預けたものを返してもらうを選びます。すると「魔法をかけ終わるまであと3年かかりますがそれでもいいですか？」と聞いてきます。もちろんここでも「OK」を選択してしまいましょう。さあ，魔法屋を出ます。もうこれで魔法はかかっているのですから。ウソー！ といって信じられない貴兄は長老の家で鑑定してもらって確かめてみましょう。

た，ただしです!! ここに大きな落とし穴があるのです。たとえばHEALの魔法をかけようと思って，この方法で木星，太陽とかけてもらっても，冒険中にHEALの魔法は使えません。「えぇぇ，どうしてっ，ちゃんと木星，太陽が宿っているのにぃ！」といわれそうですが，使えないものは使えないのです。「じゃぁ，最初の方法はパラメータを上げるときにしか使えないのかぁ」と思ったあなた，それはアマイ。たとえば，SUN RAYをかける場合を考えると，火星，太陽，木星は最初の方法（3年待たない）でかける。そして最後の水星を普通にかけて(要するに3年待つ)もらえば大丈夫。冒

険中にはちゃんとSUN RAYの魔法が使えます。「それじゃ，この方法をとればどんな魔法も3年でかけられるわけか」と悟ったあなた，それで正解。よく，理解できなかった人はもう一度読み直してみてね。これを知っておかないと，ソーサリアンはとても一世代で全シナリオをクリアできるほどあまくはできていないのですよ。

ところで，さっきのSUN RAYですが，これは火星，木星，太陽，金星が宿れば使えるといったのに，かける魔法は火星，太陽，木星，水星だなんておかしいじゃないか，と思ったあなた，スルドイ。これはどういうことかというと，魔法の要素は互いに強めあったり，打ち消しあったりするのです。具体的にいうと，土星のかかっているアイテムに土星をかけると土星×2となりますが，月に火星をかけると打ち消しあって消えてしまうのです。あるいは木星のかかっているアイテムに水星をかけると，木星と火星が宿ってしまうというぐあいなのです。このへんが魔法をかける難しいところですね。

ようやくシナリオに到着

さて，やっとシナリオの話になるのですが，このゲーム，シナリオが15個あることは先に述べましたね。果たして先に発売された88版とシナリオは一緒なんでしょうか。88版を全部解いたと，先月から自慢して歩いている吉田賢司君18歳恋人募集中（組曲「Ys」で一緒に仕事をした人です）に，ここで聞いてみることにしましょう。

「おぉい，吉田君，88版とX1版との違いはなんなの？」

「音楽の一部と表2どぁー」

といって彼は去って行きました（変な人です彼は）。そう，88版とは明らかにどこか違っているようです。それは表2を見てもらうとして，私も先日やっと全部のシナリオを解き終えたので，これからいくつか取り上げて，その一部を教授しましょう。

シナリオⅠ（レベル1）消えた王様の杖

これは誰もが最初にやろうとするシナリオですね。でも，このシナリオは結構難しいと思います。マップがレベル1にしては広いので，どの穴（扉）がどこにつながっているかをメモったほうがいいかもしれません。それと隠れアイテムの位置が88版と違っているようです。

そうそう，いい忘れました。このシナリオのように，アイテムを持ち帰るようなことを目的とするシナリオがいくつかありますね。目的のものを持って帰ると，そのアイテムを献上するか聞いてきますが，別に献上しなくても構いません。そのアイテムが装備できるものであれば，身に付けて次の冒険中に使うこともできるのです。別にKRMが減ったりしませんよ，ただ，気持ちがスッキリしませんから，1回目はちゃんと献上してもう1回そのシナリオをプレイしたときは自分のものにしちゃうというのが正しい楽しみ方だと思います（同じシナリオは何度もプレイできます。念のため）。

Ⅰ（レベル2）失われたタリスマン

このシナリオに出てくる教祖が倒せない人は，まだキャラクターが弱いということです。レベル1のシナリオで修行を積みましょう。ちなみに，このシナリオのデカキャラ「サンドマリボー」は，なかなか簡単には出てきません。

Ⅰ（レベル4）呪われたオアシス

ひたすら疲れるシナリオです。同じ場所を何度も行ったり来たりしなくてはいけないという……。なんで油壺はいっぺんにたくさん取れないんだっ！

Ⅰ（レベル5）盗賊たちの塔

これは，かなり難しい。シナリオ中盤，地下水路に落とされますが，あの水路にはトラップがありますよ。この地下水路には後半にもう一度行くことになります。なぜ行くのかはプレイしてればわかってきます。

シナリオⅡ（レベル1）暗き沼の魔法使い

このシナリオでは一部マッピングが必要です。それからアイテムを見つけたからといって全部取っちゃうと，あとでたいへん。バシバシ2段ジャンプを使って動き回れば，ピコッと取れるはずです。

Ⅱ（レベル2）ロマンシア

これは簡単。ロマンシアのオリジナル版とは比べものにならないほど簡単になっています。モンスターはバシバシ殺してもKRMは減らないし，陰険なニセ物アイテムもないし。突然ですが，病気を治す仕事なんかは絶対にプロに任せるべきですよ。えっ，プログラマのことかって？　それはあなた，病気になりやすいプロのことでしょ（意味不明の会話を交わしつつ次へと進むのです）。

Ⅱ（レベル3）紅玉の謎

これはたくさんある鳥の巣のなかを見て回るのがひとつのテでしょうね，きっと。

Ⅱ（レベル5）呪われたクイーンマリー号

このシナリオはとにかく凄い。国産RPGのなかじゃピカイチのシナリオでしょう。これ単体で売っても，恥ずかしくないくらいよく練られています。聞き込みをして殺人犯人を捜すところなんか，まるで「マンハッタン・レクイエム」みたいだけど，これが後半になって，悪魔が関係してきたりして……。



死者復活の実験を阻止する「氷の洞窟」



伝説の聖水を求めて旅立つ「不老長寿の水」

いままでファンタジーRPGというと，ダンジョンのなかをさまようのが普通だったけど，このシナリオはなんといっても舞台が船中というのがいい。

ただし，このシナリオは長いから初めてやったときは3～4時間かかりました。このシナリオに限っては，腰を据えてじっくりやったほうがいいでしょう（人にプレイして見せたら寝てしまう人まで出た，というほど長い）。

シナリオⅢ（レベル1）天の神々たち

このシナリオはいちばんやさしい（と思う）。経験値稼ぎにはもってこいです。ちなみに，このシナリオで手に入る「神の腕輪」はなかなか使えるアイテムですよ。初めてプレイする人は「消えた王様の杖」をやるよりこっちのほうがいいかもしれません。

Ⅲ（レベル2）氷の洞窟

ここはけっこう難しい。「このシナリオにはFLYの魔法がないとだめなのかなぁ」なんて思わないように。FLYを使わないと解けないシナリオなんてものはありませんよ。FLYはほとんど「トラップからの脱出用」，または「隠れアイテム探し用」の魔法ですから。

Ⅲ（レベル3）メデューサの首

このシナリオは，私は「呪われたクイーンマリー号」の次に好きです。

廃坑のメデューサは初めからお目にかかれますけど，絶対倒せませんよ。でも一度

会わないとだめなんですよね，これが。そして廃坑の天井から声が……。

Ⅲ（レベル4）囚われた魔法使い

な，なにもいいません。このシナリオに関しては。だって最後が……。

Ⅲ（レベル5）不老長寿の水

このシナリオ，レベル5にしては簡単。ただし，このシナリオの最強（？）の敵ドッペルゲンガーには気をつけろ！　いつの間にか，パーティのキャラクターが敵にすり変わっているぞ！　だから，このシナリオをやるにはある魔法が絶対に必要不可欠なのです。

いやー，音楽はさすがです

このソーサリアンのシナリオディスクⅢって，音楽が88版より少ないみたいだなぁ。ディスクの容量の都合かなぁ。そう，音楽といえば，X1turbo版ソーサリアンの音楽は凄い。ゲーム中にFM8声＋PSG3声＝11和音を鳴らしているんですよね。多少曲も88版のものからアレンジされていたり，どうやら88版にはない曲まであったりして全部で60曲近く入っているらしいので，ぜひ，ステレオにつないで聞いてみてください。

あ，そうそう，ミュージックモードを見つけた方，絶体に「西川善司のソーサリアン，ミュージックモードめっけ係」までお便りください（私はいかなる手段を使っても，このミュージックモードのありかを知りたい）。

そのほか魔法の情報，ハーブの情報，または，このシナリオが解けないよう，といった内容のお便りもこの西川善司はお待ちしております。

とにかくこのソーサリアンは，早解きはなるべくやらないようにしましょう。1つひとつのシナリオは短く，そしてそれほど難しくもありませんから，別に早く解いても偉くもなんともありません。それより，冒険中のキャラクターになりきり，「剣と魔法の世界」を思う存分，楽しんでくださいね。きっと新しいRPGの世界が見えてくるはずですよ。

それではまた来月。

表2　吉田君のシナリオ採点表

シナリオ名	88版との相違度	難易度	理想的なプレイ順	面倒くささ
シナリオⅠ				
消えた王様の杖	×	3	3	4
失われたタリスマン	×	3	6	3
ルシフェルの水門	×	4	9	4
呪われたオアシス	×	4	10	5
盗賊たちの塔	○	5	15	5
シナリオⅡ				
暗き沼の魔法使い	△	1	2	2
ロマンシア	○	1	4	1
紅玉の謎	○	2	7	2
暗黒の魔道師	○	4	12	4
呪われたクイーンマリー号	○	5	14	5
シナリオⅢ				
天の神々たち	○	1	1	1
氷の洞窟	○	3	5	3
メデューサの首	○	3	8	3
囚われた魔法使い	○	2	11	4
不老長寿の水	○	2	13	4

88版との相違度　×：まったく違う　△：一部違う　○：まったく同じ
難易度　難しい　5 ←→ 1　やさしい
理想的なプレイ順　吉田君の独断と偏見によるもの
面倒くささ　面倒くささとはシナリオを解くのに，どれくらいの時間がかかるか，ということも表しています
面倒くさい　5 ←→ 1　単純である
（時間がかかる）　（比較的すぐ終わることが可能）

表1　魔法の作り方（ほんの一部）

この表の見方：表中に書かれている数字は要素をかける順番を示しています。○の部分は順番を伏せてありますから，自分でその順番を探してみてください。

魔法名	火星	水星	木星	月	太陽	金星	土星	便利度（1→5）
ANTI-MAGIC			○	○				1
ASTRAL FIRE		○		○		○	○	
ASTRAL WAVE	1			2		3		
AIR SLASH		1	3				2	
BARRIER		1，2		4	3		5	3
CHANGE AIR			1	2		3		4
CORROSION		2		1			3	
CURE					2	1		3
D-CORROSION		○	○	○	○		○	
DEATH	○			○	○	○	○	3
DEG-DEATH	2			3	1	4		3
DEG-FIRE	○	○	○		○		○	
DEG-NEEDLE		3			4	2	1	5
DISPEL	○		○	○		○		
EXIT	1	3			2			
EXOCISM				3	2	1		3
FIRE FOX		1，2		4	3			
FLAME	2				1			
FREEZE	2				1		3	3
GOD THUNDER	○	○	○	○	○	○		
HEAL			1		2			5＋α
HYPNOTIZE	2		1			3		
ICE WALL		○	○	○	○			
ILLUSION	1					3	2	
JET STORM	○	○	○	○		○	○	
LOST MORALE				4	3	2	1	

魔法名	火星	水星	木星	月	太陽	金星	土星	便利度（1→5）
LIGHTNING				1		2		
LIGHT CROSS		1，4	3		2			5
MELT					3	1，2	4	1
METEOR	3	4			1，2			3
NAPALM			2	5	4	3	1	3
NEEDLE	1			2				3
NEGATE	1	2					3	2
NOIRA-TEM	○	○	○	○	○	○	○	5
PEACE		2				1		
POISON	1						2	
PROTECT			1	4		2	3	
RESURRECT			1		3	2		5
ROCK RAIN	○		○		○	○	○	
RESOLUTION		2			3	1	4	
SAND STORM		1	2					
SCARE			2				1	
S. EXPLOSION	○		○	○	○		○	
SPARKS	1					2		
SHIELD				2	1			
STILL AIR		1			2		3	
STONE FLESH			2		4	3	1	3
STORM	1	2						2
SUN RAY	1	4	3		2			5
SWOON	○	○	○		○			2
X RAY			4	2	1		3	3

THE SOFTOUCH

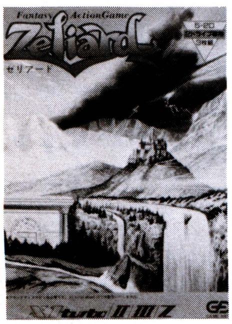
●ゼリアード



行くぞ洞窟探検隊，クリスタルめっけ

Shimizu Kazuto
清水　和人

石にされたお姫様を救うため，我らが清水和人は地下にある20もの洞窟を旅するのだった。果たしてそこにはどんな怪物たちが待ち受けているのだろうか。いま9つのクリスタルを求めて壮絶な戦いが地下迷路で展開されようとしている。



X1turbo用　5"2D版3枚組　7,500円
（model10不可，2ドライブ専用）
ゲームアーツ　☎03(984)1136

ひと昔前のRPGは退屈なものだった。限りない敵とのバトルで雀の涙ほどの経験値を得，わずかな金と食料を持って砂漠のような単調な地形をただひたすら旅しなければならなかった。

しかし，いまは違う。あたかもひと昔前のAVGのように，変化と波乱に満ちた目まぐるしい戦いと自分自身の推理力が必要な時代へと移り変わってきたのである。いつの間にかこんなに進歩し，RPGはパソコンゲームの一角を堂々と担うようになってきたのだろうか。今回，これからご紹介する「ゼリアード」も，その昨今のRPGの流れからして決してRPGという名に恥じない変化に富んだゲームなのである。

ゼリアードは，RPGのなかでもリアルタイムRPGのジャンルに属し，マップのなかを移動し，そこで出会った人々から話を聞いたり，敵が登場すれば一戦交えることになる。まあ，お馴染みのハイドライドやザナドゥ（画面が縦だからどちらかといえばザナドゥに近いのかな）のパターンというわけだが，そこはそれ，ほかでは見ないような工夫が随所に折り込まれているので楽しませてくれる。それではその世界を覗いてみることにしよう。

見事なイントロ

ゲームに入る前に，ストーリーを紹介した紙芝居（絵が出てきてはストーリーが文字によって説明されるといういつものやつ）があるのだが，このゼリアードはそれ用のディスクが1枚用意されているという凝りようだ。ずいぶんと長いオープニングで，はやる気持ちをじらされるわけだが，2回目からは見なくてもいいようになるので最初はよーく見ておいたほうがいい。

まあ，簡単に要約すればお姫様が石に変えられちまったんで，それを元に戻すために9つのクリスタルを集めてくれば，めでたしめでたしということなのである。ちなみにところどころでしゃべってくれるのはいいのだが，お姫様は本当に鈴のような声をしているというのがわかった。そういやぁ，ゲームのいっちゃん最初にごちゃごちゃ聞こえたような気がしたが，よく考えると「Presented by GAMEARTS」といっていたらしい。

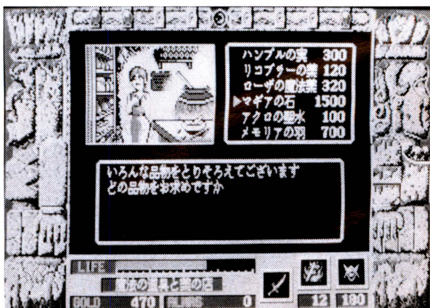
で，オープニングが終わるといよいよゲームスタートだ。舞台はフェリシカ姫の住んでいるフェリシカ城から始まる。そうしてザナドゥのようにムララの町に行って買い物をするってぇ寸法だ。ただ，町をうろついていると，そこにいる人間たちがいろんな情報を教えてくれる。このあたりは，ストーリーへの気配りが感じられて嬉しい。

さて，そのムララの町情報のなかで重要になってくるのが，洞窟が8つあって怪獣がいるってぇことと，次の目的地は洞窟のなかにある「サトノ」という町らしいということだ。さらには音のする扉があって，そこには魔物がいるってぇことだ。つまりは，その扉のところで魔物をやっつければ次の洞窟に行けるという寸法らしい。

まずは，町の商店街で買い物である。この私は買い物上手として一般世間に広く知られているが，まずは“賢者の楯”を買った。“賢者の剣”というのもここでは売っているが，この剣は1500ゴールドと高いし，“修行の剣”というのを最初から持っているから，とりあえずはそれでガマン。賢い消費者なのである。そのあと“セルドの薬”をいくつか買って残りのお金を貯金してからスタートしよう（もしやられてしまうと復活するときに持ち金は全部取り上げられてしまう）。

洞窟のなかに入ったらとりあえずはボカスカ敵をやっつけるのであるが，この敵がなかなか手強い。最初の洞窟はマリシア洞というやつなのだが，ここではカエルさんがいやな存在。だから間合いをはずして，ピョンピョンと跳ねて向かってくるところをグサッとやるのがいい（ほかの洞窟には炎を吐くやつもいる）。コウモリもいるが，これはうまくやっつけると10アルマスが手に入る。このアルマスは金に換えることができ，ムララの町では1アルマスは4ゴールドになる。普通，怪物は1アルマスしか持っていないが，カエルとコウモリは10アルマス，またマリシアの洞窟の奥深くにいる背の高いやつも10アルマスになる。ここで稼いで，“賢者の剣”を買おう。

やられそうになったら，“セルドの薬”を飲む。それがなくなったら地上へ引き返す。そして地上でやらなければいけないことは，1）賢者に経験を見てもらう。2）銀行でアルマスを換金する。3）武器屋で楯を修理する。



店に入るとカワイイ声で迎えてくれるお嬢さん

4)教会で休む(これは無料)。5)銀行に預金(郵便局では貯金と呼ばれている行為である）する。6)賢者のところに行ってセーブする。といった順になる。もっともこの順番は町によって違う。体力が5のレベルになると，ムララの町の賢者アモーニはもうレベルアップしてくれなくなるので，ようやくそこで例の怪獣の扉を開けよう。洞窟内はマップを作らなければならないほど複雑ではないが，ここに行くとあそこに出るといったメモくらいは書いたほうがいいみたい。

我は行く，さらばムララよ

例の扉のカギを開けると，デカキャラがポンと出てくる。しかし，それが終わるとこの面は閑散としている。かくしてすんなりテンポよく，サトノの町に出られたのであった。このゲームでは，同じ面にいつまでも苦しめられるということは少ない。だからこのテンポはRPGのなかではたいへん心地よい。

サトノの町にもひと通り店が揃っている。まず賢者ビュエナのところへ行くと，最初の魔法“エスパーダ”がもらえる。薬屋では魔法が回復する“リコブラーの薬”や楯を回復させる“アクロの聖水”を売っているが，たいしてこれは役に立たない。ここではなんといっても最強の武器“マギアの石”を手に入れるべきである。

この“マギアの石”は，体の周りを石がグルグル回って，防御と攻撃を同時にやってくれるという，宇宙戦艦ヤマトのアステロイドベルトみたいなやつだ。これを身に付けて走り回れば簡単にお金持ちにもなれる。

また，サトノの町での疲労回復は宿屋に泊まればいいのだが，これは30ゴールドと有料（まあ,安いけどね)。ある程度お金がたまったら2980ゴールド（ニッキュッパなんて，まあオシャレー）の“防石の楯”を買おう。

ここらで剣の使い方を教授しておこう。ジャンプしたり，しゃがんだりしながら使うのはもちろんのこと，テンキーの8と2を同時に押して剣を縦に振り下ろすことや，ジャンプして降りてくるときテンキーの2を押して剣で下の敵を突く方法などは覚えておいたほうがいい。ちなみにキー操作は非常にスムーズだ。

次のペリグロ洞の奥には，「開けるな」といわれている箱があるが，かまわないから開けてしまえ。なかにはコウモリみたいなスレイヤーがいる(こいつは黄金バットか)が，やっつければ体力が元に回復するのだ。まあ，やられたらセーブしたところからやり直せばいい。げに諸行無常よのう。

でもって，この世界の親玉は大ダコのプルポ君（名前だけはカワイイ）だけど，またまた弱っちいデカキャラだったりするわけだ。こんなのにやられているようでは，勇者と呼ばれるのはチャンチャラおかしい，となるわけである。

我はもっと行く，さらばサトノよ

プルポ君と戦ったあとは，マデーラ洞に出る。が，そのすぐ左下あたりにはボスクエの村がある。ここでは，疲労回復にお役に立ちます薬の“ハンブルの実”や，6800ゴールドの“精霊の剣”を売っている。新しい魔法“セーダー”は射程距離や威力がグンとお得な攻撃用魔法である。そしてなんといってもおいしいのは，1アルマス6ゴールドという換金率。ここがほかの洞窟に比べていちばん良心的なわけだ。先の洞窟に行った勇者も必ずここに戻ってきて換金するという，伝説の銀行がこの村には存在するのである。

ほかの町でもそうだが，魔法屋さんでは女の子が「いらっしゃいませ」とブリッ子な声で迎えてくれる。武器屋のオヤジはなにもしないで出て行くと「ひやかしなら出て行け」と机を叩く。宿屋の主人は泊まるとき「はい，1名様ご案内」といってくれる。このようにそこいらじゅうでよくしゃべったりするのは，さすがゲームアーツ。

さて，ボスクエから行くマデーラ洞，ライズ洞は樹の世界になっていて，ここではいちばん太い樹の根元にある“勇者の紋章”を取らないことには先に進めないらしい。この世界あたりからメモを細かく取らないと，道に迷ってしまうかもしれない。

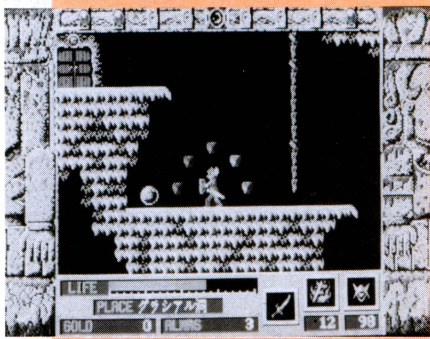
そうして紋章が手に入ったら，今度はボスクエの村に行って番兵にそれを見せるといい。するとなかに入れてくれて，巨大な火の鳥みたいなかっこしたデカキャラ・ポローがいる。それにしても，弱っちいやつ。

我は行くんだってばあ

ポローと遊んであげたあとにたどり着くのはスケートリンク，井上陽水の氷の世界だ(ちょっと古かったかな)。走っていて止まろうとするとツルリと滑ってガケから落ちたりするという，たいへんいやらしい世界なので，ガケの手前は要注意。エスカーチャ洞とグラシアル洞を経て，ヘラダの町に出るのはこれまでになく困難なので，危なくなったら引き返してボスクエの村にすかさず戻ったほうがいい。ヘラダの町ではあの“マギアの石”を売っていないので，ボスクエの村で買っておいて，ヘラダで一度売っておこう。そうすれば次からヘラダでも買うことができるようになるのだ。



勇者ゆえの悩みもあったりする



ボコボコと敵をやっつけるマギアの石

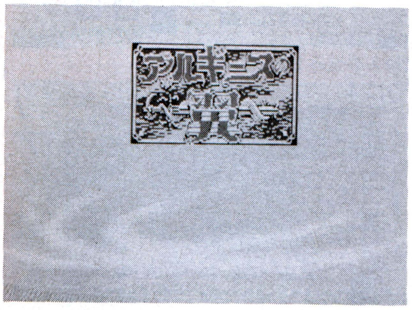
ここでの新しい魔法は炎の“フェーゴ”だ。宿屋は70ゴールドと暴利をむさぼっているし，1アルマス2ゴールドと換金率の悪さは最低。“栄光の楯”が9800ゴールドするので，これなんかとても買えないや。

ここでの目的は，“ルゼリアの靴”を探すこと。これは目の前に見えているくせに取れないという，とんでもなく厄介な場所にある。そうそう，この面からいよいよ本気にならなければならないような雰囲気が漂ってきた。なにはともあれ，方向オンチの私はすぐに迷子になってしまうのでありました。

ここから先には，ようやくその本性を発揮してくる強っちいデカキャラが待っているし，壁を壊さないと通れない場所や，一方通行のバリアが張ってあるところもあったりもして，だんだん難しくなってくる。こうした進行は決して単調にならないように次々と新しい展開を見せてくれて非常にいい。

結局，このゼリアードはトントン拍子にプレイさせてくれて，終わったあとにさわやかさが残るゲームだといえそうだが，そうかといって簡単には解かせてくれるわけではない。このあたりの塩加減はとっても苦労して作ったんだろうなあ。で，肝心なお味はというと，とってもマルでした。

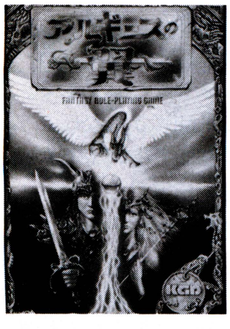
●アルギースの翼



これこそ工画堂のRPGなのぢゃ

Kuramochi Ryouichi
倉持　亮一

SFに中世のスタイルをマッチさせた新しいタイプのRPG，それが「アルギースの翼」だ。久びさに登場した工画堂の最新作に，これまた本誌初登場の倉持亮一が，平和を守る戦士となって，悪戦苦闘の妖獣退治の旅に挑戦する。



X1 turbo用　5″2D版2枚組　7,800円
（ただしFM音源ボードがあればX1でも動作可，2ドライブ専用）
工画堂スタジオ　☎03(353)7724

プロローグ

宇宙の遙か彼方，第5銀河星団α星にアルギース王国と呼ばれる国が栄えていた。もともとこの国は天空界にあったが，やがて地上界にも安住の地を求めて人々が移り住むようになった。そしてそのころからアルギース王家には代々双子が生まれるようになり，その2人の世継ぎが天空界と地上界をそれぞれ治めることを世襲としたのである。

しかし，地上界には以前から妖獣が生息していたのだが，それを地上界の王は民衆と地，水，風の3人の精霊とともに力を合わせて制圧し，平安の地として築き上げてきたのだった。そうしてアルギース第15代のとき3人の精霊に命じて天空界の扉を閉ざしたのであった。不可解なことに，天空界の扉を閉じたあと，アルギースの王家にはもはや双子は生まれなくなったのだが，あるときひとりの司法長がこういった「アルギースに異変が起きるそのとき，再び双子が生まれ，平和に向けて立ち上がるであろう」と。

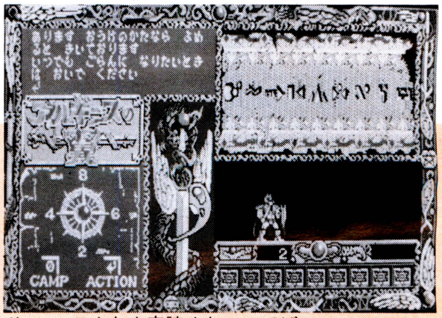
そしてアルギース第23代のとき，その言葉は現実のものとなった。過去に滅ぼしたはずの妖獣たちのなかに高い知能を持った頭目が現れ，再びアルギースの国を脅かし始めたのだった。ついに国王と妃は殺され，城は廃墟と化した。民衆はもう妖獣たちに立ち向かう気力も失せ，ただおびえたままの日々を過ごしていたが，両親を殺された王子は，悲しみを胸にただひとり波乱に満ちた冒険へと旅立つのだった……。

＊　＊　＊

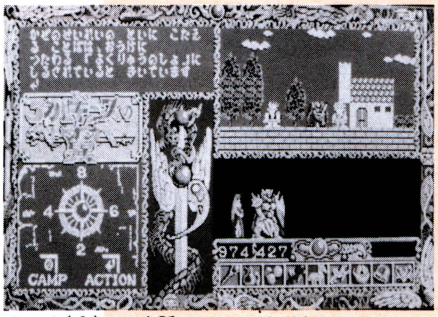
毎度シナリオには定評のある工画堂だが，しばらくのご無沙汰はあったものの相変わらずストーリーは魅力的だ。特にラピュタ大好き人間にとっては「天空」という言葉を聞いただけでワクワクしてしまう。やはりこれからのRPGのヒーローはいつまでも地ベタにへばりついていてはいけない（ゼリアードは確か地下が舞台だったな）。脳天気なヒーローはやはりスーパーマンのように地上を離れて活躍すべきなのだ。と，張り切ったのまではいいけれど，この私の意に反してこのゲームはどんどん地上を舞台に展開してしまう。やがては天空界にも行くことになるのだが，結局はまた地上界に舞い戻ってしまう。せっかく魅力的なシナリオだと感心してあげたのに，感心度の半分は返してもらおうっと。

さて，ゲームはフィレスの町から始まる。地上界は大きく分けて中央のマップを中心に東西に分かれていて，ちょうどまん中に位置するこの町は薬屋やHPを回復するための休憩所，セーブを行う宿屋などがあるので，当面はここが拠点となる。アルギースではあまり買い物に重点が置かれていないので，店はあまり出てこない。売っているものもせいぜいHP満タンにしてくれる薬か，そのうち相棒として活躍してくれる鷹のエサを売っている店くらいのものである（この鷹は毎回エサをあげないと，次から働かないというわがままなやつだ）。

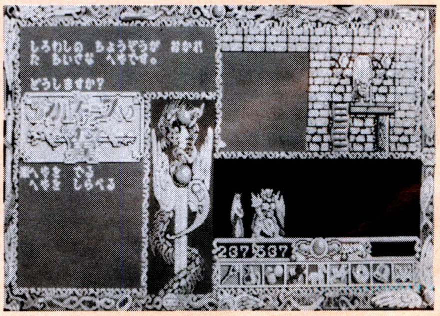
武器屋はあるのだが，どこも品薄で剣，鎧，楯の3点セットが2つの町で売られているだけ（この2つはレベルが異なっているけどね）。まっ，なんでも揃っているコンビニエンスストアが2軒登場すると思えばよろしい。日頃から「なにが悲しゅうて，世界平和を取り戻そうとする正義のヒーローがせこせこ金を集めて武器を買う算段をしなけりゃならんのだ」と思っているこの私にとって，このショッピング感覚は買いである。



ひぇー，こんな文字読めないってば



見てよ（？）この会話。ストレートでしょ



苦労して塔の上まで来たのに，扉は開かんのか

いやー，皆さんほんとにご親切

旅立ちの町，フィレスの人はみんな親切だ。いやフィレスに限らずどの町の住人も，たったひとりで妖獣退治に出かけようなんぞという物好きには愛想がいい。なにがどこにあるか，なにをどうするべきかなどをハッキリと教えてくれる。たとえば，

「この町の北のほうに鷹匠の町があります」

へえー，そうなんですか。

「“水の木馬”を使うとき，西の地への道が開かれるといわれています」

そりゃどうも。

「地の精霊はセルリノ山脈の中腹にある大きな木の前に現れると伝えられています」

こりゃ，大切な話をご丁寧に。

「南の山岳地帯が……」

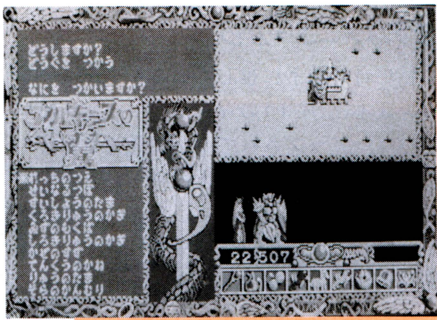
あのー，もう少し回りくどくいってもらえないでしょうか。どうもストレートすぎちゃって，私はもう少しナゾを含んだヒントみたいなのを期待してるんですけど……。てな，ぐあい。昔からのことわざにもあるでしょ「親切はゲーマーのためならず」って（ちょっと違うような気もするが）。少しは遊んでる人間のことを考えてほしいものである。

まあそれはいいとして，町を出ると画面の右上に有視界マップが広がる。そこをトコトコと歩いて行くと突然ディスクがカラカラと回り出し，画面右下に妖獣がご登場である。ここで「戦う」コマンドを選ぶと，工画堂お得意の戦闘モード（突然だがサイキックウォーはどこへ行ってしまったんでしょ）になる。戦闘中はただ高見の見物，「ホレいけっ！」，「ガンバレ」とひたすら主人公の応援団をやってればいい。

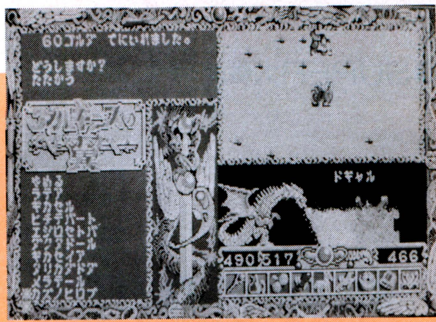
ただ，中盤戦ともなってくると「逃げる」とか「鷹を飛ばす」などと緊迫した雰囲気を忘れて悠長なことも簡単にできるようになるので，比較的町から町への移動には苦労しなくなる。地上マップではスクロールのスピードも速いので，アッという間に目的地に到達することができる。

アレッ，修行って簡単なのね

オーグマットという町では，またまたご親切なことにいきなり象形文字で書かれた古文書を見せられた。「王家の方なら読めると聞いています」だと。読めないつーの，そんな文字。でも，ここで活躍するのがこのゲームのオマケに付いている「翼龍の書」という立派な装丁の虎の巻。ここにはアルギース前史，扱える魔法，精霊を呼び出すときの呪文，古文書の正しい読み方（本当



城のなかに入るのがまたひと苦労



この魔法って怖いくらい強力なのね

はこんないいかげんな呼び名ではない）などが記されている。とにかくこの本はオマケと呼ぶにはあまりにも丁寧な作りだし，古いイメージを出すために紙質も凝っている。おまけにハードカバーに金の刻印とくりゃ，誰が見てもカッコイイ。

まっ，それは置いといて，トキルという町には攻撃力をアップしてくれるという人がいる。この人に会うと，

「攻撃力の修行をしてあげましょう。攻撃力を10上げるのに50ゴルダかかります。修行を受けますか？」

と聞かれる。新しい剣もなかなか見つからないので，ここで攻撃力でも上げておこうと50ゴルダを支払う。さあ，修行だ。すると突然，

「修行は終わりました。また来なさい」

エッ，もうおしまい。あれま，なにかしらのイベントがあると思ったら，たったひと言で終わりなの。どうも，また来なさいの言葉の裏には，「お金を持って」という接頭語が隠されているような気もするが，この修行は何度も受けるハメになるので，どうやらそのたびにイベントなんぞやってられない，ということらしい。でも，もう少しだけ「俺は強くなったぞ」という自信が持てるようなことが儀式としてあってもいいんじゃない。

いつまでもお金を使って修行をしているわけにはいかないので，トキルの町で聞かされたセルリノ山脈へと向かう。ゲーム中に流れるBGMは軽快だ。FM音源対応のサウンドは，残念ながら８重和音を使いきってはいないが，聞いていると，ちょっと毛色が違っていてとても楽しい。ただ，ゲーム構成とぴったりマッチしているかというと少々疑問だが，曲自身はよくできていて，従来のRPGの音楽とは一線を画している。特に町のシーンに流れる音楽は構成が新しいので，「俺は音楽にはチトうるさいぜ」と自負している人には，ぜひとも聞いてみてほしい。

さて，セルリノ山脈にある大きな木のところであれこれやっていると，突如画面がピカピカッとフラッシュして地の精霊が現れた。

「アルギースの若き王子よ。よく参られた。アルギースの大地の嘆きはわしの耳にも届いておる。これなる“裂地の杖”をそなたにつかわそう」

おいおい，自分のこと「わし」なんて呼ぶなよな。どう見たってあんたは若い女性のカッコしてるんだから。まっ，そんなこんなでここから険しい山に囲まれた東の地へ行くことになるのである。

残念，ハワイ旅行にあと一歩

このアルギースには，戦闘を有利にするアイテムのようなものがほとんど存在しない。途中で手に入れることができるアイテムは，すべて次の地へ行くための通行証のようなものばかり。そしてこれらのアイテムは，順序よくつながっているから，決してプレイ中に惑わされることもなく，最終目的地へと向かうことができる。

しかし，その素直さは認めるが，逆にいまひとつインパクトに欠けるような気がする。アルギースの各地に点在する塔や城には次の場所へのヒントが隠されていて，そのために迷宮を歩き回ることになるのだが，その迷宮はそれほど複雑ではなく，面倒なマッピングも必要とはしない。これはたいへんユーザーフレンドリで，結構である。ただ，そこにピリッとくる味付けがもう少しほしかったのも事実。

シナリオというものはAVGやRPGにとっては要となる部分だし，シナリオによってゲームの面白さが左右されることは，誰もが認めることだろう。しかし，完成されたシナリオにはプレイヤーをワクワクさせてくれるような演出が伴わなければ，せっかくのシナリオのよさも生かされない。

このアルギースの翼というゲームは，脚本，美術，音楽のどれを取ってもたいへんよくできているのに，演出家の努力がいまひとつという気がして少し残念だった。惜しいなあ，あと一歩でハワイ旅行だったのに……。

THE SOFTOUCH

●SUPER大戦略



いま熱く燃える勝利への長い道

Kageyama Hiroaki
影山　裕昭

前作はただの序章であった。このSUPER大戦略こそ本当の戦場なのだ。生産タイプや４カ国モードも加わって，さらにパワーアップしたSUPER大戦略がいま我我の前に姿を現した。さあ，熱い男の戦いがここから始まる。



X１turbo用　5"2D版2枚組　8,000円
(Model10は要G-RAM，2ドライブ専用)
システムソフト　☎092(714)6236

まずはメニューから

私が自分勝手に推しているシミュレーション３本柱(注１)のなかから，今回大戦略がパワーアップされてX1に発売されました。初代大戦略は現代大戦略という名前で３年前に98に発売されて以来，88，X1，FM，さらにはMSX2に移植され(注２)，爆発的人気を得たソフトです。そのパワーはまさに親の遺言級でした（わかる人は古くからの読者ですねぇ。わからない人はバックナンバーを見よう)。

その後98には大戦略パワーアップセットや大戦略IIが，X1などにはマップコレクション(注３)が発売されましたが，大戦略IIによって戦術が増えた98版と比べると，X1版はただマップが増えただけで物足りないところがずいぶんありました。そこで，今回発売されたSUPER大戦略ですが，できるだけ大戦略IIの仕様に近づけたようで，ここまでよくやったなぁ，というのが素直な感想です。どこがどう変わったかは大戦略ファンの人なら興味津々ですね。それでは，ぼちぼち始めてみましょうか。

「すいませーん，SUPER大戦略ひとつください」

いただきます

まずは，前作からの変更点をズラッと並べていきますから，息切れしないでくださいよ。ゲーム画面を見て最初に気づくのが，ビュースクリーンが大きくなったことです。これによって前作で捕らえにくかった戦況が，ひと目で確認できるようになりました。マップサイズも64×64(前作は40×40)となり，前作よりかなり大きくなっています。それにつれて，ユニット数も48個に増えたのですが，まだちょっと足りません。ホントは60個ぐらい作れるといいんだけどねぇ。

それとメッセージはすべて日本語になって，とってもフレンドリ。それから，移動の方法もずいぶん変わりました。移動させるユニット上でF1を押すところまでは前作と同じですが，ここから先がちょっと違います。SUPER大戦略ではF1を押すと移動できる範囲が色違いで表示されるようになったのです。おかげで自分でどこまで動けるか考える必要はなくなりました。プレイヤーは目的の移動場所にヘックスカーソルをもっていってスペースキーを押し，リターンキーで決定→移動，というようになりました。

これについてはつまらなくなったという人も何人かいるようですが，やっぱりこのほうが便利，初めて遊ぶ人でもすぐ理解できそうです。

お次は生産タイプ。これはSUPER大戦略になってから新しく設けられました。このことは，あとで詳しく話しましょう。ほかにも，４カ国までプレイできる，画面全部を使う派手な戦闘シーン，要塞，湿地，橋の３つの地形が新たに追加，収録マップ数が25個になった(前作は16個)，生産できるユニットの種類が121個(すごい数！)になった，などSUPERの名に恥じないものになっています。このなかでひとつだけ悲しい変更点が，turbo専用になってしまったこと。メモリ容量の関係(turboになって増設されたG-RAMにデータを置いている)でこうなってしまったんだろうけど，福岡のシステムソフトさん，全国のX1ユーザーはいつまでも待ってますよ！

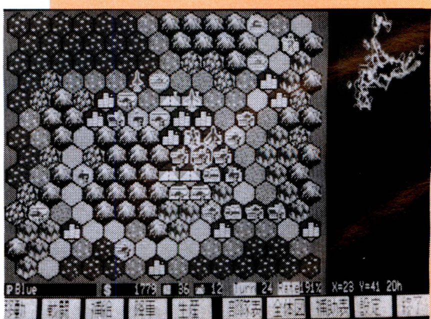
What生産タイプ？

SUPERになって最大の変更点は，４カ国で遊べることと，生産タイプの設定といえます。２カ国でプレイするのと違って３，４カ国ともなると，最初にどこから攻めるのかを決めるのが重要になります。かとい

注１　いわずと知れた，三国志，信長の野望，大戦略のことである。
注２　秋ごろにはファミコン版も発売されるらしい。それにしても四角いヘックスもやはりヘックスと呼ぶのだろうか。
注３　X１(88兼用)，98，ともに限定発売された。ちなみに収録マップ数はX１が16個，98が45個だった。



豊富に用意された生産ユニット



うわっ，敵がここまで来ちまった

って，攻めるのに夢中になっていると，よその国が自分の首都を狙ってきます。

てなわけで，前作よりも高度な戦術が要求されるのです。いい換えれば，前作は序章であって，これこそ真のストラテジックシミュレーションなのです（ストラテジックとは「戦略上重要な」という意味）。

さて，お次は生産タイプ。これはなにかというと，前作のように生産できる兵器が決められているのではなく，プレイヤーは自国の兵器を12の生産タイプ（注４）のなかから選択することによって決めることができるのです。たとえば，中国なんかはメチャクチャ兵器は安いんだけど，性能が悪いし，逆に最新鋭部隊なんかは性能はいいけど価格が高い，というようにそれぞれに個性があります。ですから「俺は日本人だ，大和魂をなめるんじゃない！」という輩は日本を選べばいいし，「やっぱり戦争は兵器の豊富なアメリカだ！」という輩はアメリカを選べばいいのです。こんな輩がSUPER大戦略で戦うと，日本vsアメリカ，第３次世界大戦が画面の上で繰り広げられるのです。さらに，３，４カ国でプレイすれば第３国の参戦もあって，ますます面白くなるでしょう。

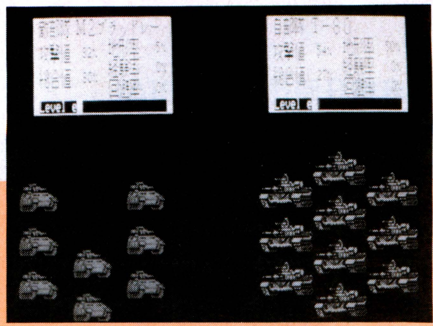
内蔵の12の生産タイプじゃ物足りない人は，生産タイプ編集でオリジナルの生産タイプを作ることができます。ここでは121種類のユニットのなかから，自分の気に入った20種類までを選ぶことができます。ここには12の生産タイプのなかには入っていない「ゲリラ兵」がいます。こいつはたいして強くないのですが，価格が＄100とお買い得で移動力も歩兵としては唯一４もあるので，中立都市がたくさんあるマップではその威力を発揮することでしょう。オリジナルの生産タイプを作るときは，ぜひ加えておきたいユニットのひとつです。

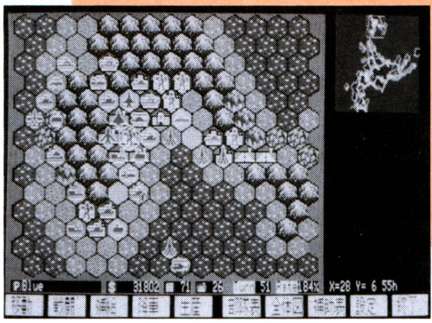
コンピュータの実力はいかに

やはりこのテのゲームでいちばん気になるのが，コンピュータの思考ルーチン。今回は私が実際にプレイしたRed Bearを例にとってリポートしてみましょう。このRed Bearは北海道をあしらった３カ国用のマップで，生産タイプは青がアメリカ１，赤がソビエト，緑が西ドイツに設定されています。マップの中心に首都を持つ西ドイツが，北のソビエトと南西のアメリカに挟まれて難しそう。アメリカと西ドイツは首都が比較的近いのですが，ソビエトと西ドイツはずいぶん離れています。私は，生産タイプはこのままに，収入率100でアメリカ１を



これが新しく加わった生産タイプ設定モード


このあとガンガンバリバリと銃撃戦が始まる


これでチェックメイト，私の勝ち



長かった戦いも勝利のうちに終了

選んでスタートしました。

最初は基本どおり，歩兵と輸送のためのユニットを作ります。歩兵は少々高くても重歩兵を作ったほうがいいみたい。SUPER大戦略では，輸送手段として戦車も使えるので，M2ブラッドレーも作ってみました。

最初はやっぱり中立都市の奪い合いです。20ターンくらいまでは，ソビエトの保有都市数が40近くあり，アメリカ，西ドイツはともに20くらいしかないヤバイ雰囲気でした。ところが，24ターン目に私（アメリカ）が西ドイツの首都を占領したあたりから形勢逆転。都市数が互角（注５）となればソビエトに勝ったも同然です。なぜなら，アメリカと西ドイツが激しく戦闘を繰り返し，アメリカが首都を占領する２，３ターン前になって，ソビエトの陸上部隊はやっと西ドイツの首都近くに迫って来たのです。

このとき，アメリカの陸上および空中部隊の熟練度がDからCだったのに対し，ソビエトの熟練度はほとんどないに等しかったのです。もう，ここからは押せ押せムード。ソビエトの都市はみるみるアメリカに占領されて，結局は51ターン目でソビエトの首都を占領することができました。

で，コンピュータの強さはどうかというと，確かに思考ルーチンは変わっています。前作にあったような，ガス欠の戦車の山は今回はできませんでした。でも「これはとんでもなく強い」という感じではありません（私が大戦略をやり込んだからかもしれないが）。特に今回プレイしたRed Bearの戦いでは，そんな印象を受けました。

ごちそうさま

私は３カ国モード初体験だったのですが，なんとか勝つことができました。中盤に，アパッチ（攻撃ヘリ）を量産したのが勝因だと自分で分析しています。それにしても，敵国の兵器の特徴を覚えるのは大変です。だって初めて見る兵器がほとんどなんですから。参考までに書きますと，51ターン終わるのに18時間くらいかかりました。試験前にはやはりこのソフトは封印しておきましょう。いまは，Fukuoka City（４カ国）をプレイしていますが，こちらはまだ５ターンくらいしかやってません。ところで，エンディングでESCを押すと，グラフィック登場の仕方が毎回変わります（20種類くらいあるのかな）。くだらないけど，やってみてください。

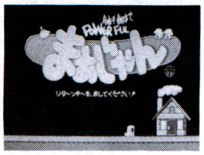
それじゃぁ，みんな，ここまで読んでまだ買っていない人は8,000円持ってパソコンショップにかけ込もう（大きな店なら１割引きかな）。えっ，ソーサリアンにイースIIも買いたいって？　うーん，こまったちゃんですねぇ。お金の余っている人は全部買えばいいけど，そうでない人には，ながぁーく遊べるSUPER大戦略を勧めちゃったりするわけなのです。

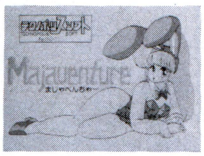
注４　アメリカ１，アメリカ２，フランス，ソビエト，イスラエル，スウェーデン，ワルシャワ，日本，西ドイツ，中国，イギリス，最新鋭部隊の12種類。

注５　首都を占領すると，その国の保有していた都市はすべて自分のものとなるのだ。

●麻雀狂時代SPECIAL
●今夜も朝までPOWERFULまあじゃん
●まじゃべんちゃー・ねぎ麻雀







おとこ度胸の麻雀3本勝負

Ogikubo Kei
荻窪 圭

巷ではいま，ギャラクシーフォースが流行っているらしい。しかし，X1/X68000には麻雀ゲームブームが突如として再燃したのだ。いまなぜ麻雀ゲームなのか。その謎に迫るべく，荻窪圭が果敢にも単身麻雀3本勝負に挑む。

麻雀狂時代SPECIAL
X68000用 5"2HD版 2枚組 7,800円
X1/X1turbo用 5"2D版 2枚組 6,800円
マイクロネット ☎011(561)1370
今夜も朝までPOWERFULまあじゃん
X1turbo用 5"2D版 2枚組 6,800円
デービーソフト ☎011(251)7462
まじゃべんちゃー・ねぎ麻雀
X1/X1turbo用 5"2D版 2枚組 6,800円
徳間コミュニケーションズ ☎03(591)9161

もう季節は梅雨です。好きだといったら電光石火の早業で後ろ指をさされる梅雨です。しかし，誰からも嫌われてかわいそうなこの梅雨は，これから暑い夏になると忙しくなって体力もお金も必要だろうから6月くらいはゆっくりと休みなさい，というお天道様のお慈悲なのです。せっかくのご配慮なのだから全天候型コートでテニスだとか，室内プールで泳ぐとか，ディスコで汗を流してナンパだとか，東京ドームを借り切って野球をするだとか，梅雨の明けた沖縄へ行って焼くなどの健康的行事は言語道断この罰当たりめ，なのです。

この梅雨時は，4人集まれば雀荘に出かけて全自動卓，大きなゲームセンターのあるところなら並んでファイナルラップもいいでしょう。3人なら，少々梅雨にしては健康的に過ぎますが，流行の波も過ぎてゆっくり遊べるビリヤードです。できればギャンブル性の強い9ボール。2人なら，8ボールや4つ玉もあります。外へ出かけるのも億劫だったりしたらスーパーレイドックやツインビーで楽しめるでしょう。

たったひとりでさみしいとき，雨で街に出るのも面倒なときはアートアンサンブル・オブ・シカゴを聴きながら，いがらしみきおを読むもよし，トーキング・ヘッズの新譜を聴きながら坂口安吾のエッセイを読むもいいのですが，閑人の頭はやはり麻雀へと回帰してくるのです。

なぜか麻雀ソフトが復活した

パソコンれい明期，「コンピュータ→高速計算→思考ゲーム」という三段論法が夢だったころ，強い麻雀の思考ルーチンを開発しようと大学の大型コンピュータで，でかいプログラムを走らせていた人々の思いも乗せて，麻雀ゲームが乱立した時代がありました。それに輪をかけたのがジャンピュータの登場。そののち，パソコンではハドソンの「雀狂」，シャノアールの「プロフェッショナル麻雀」以来これといったゲームのない低迷期が始まりました。麻雀自体が廃れたのか……。

否，否，否。確かにいまの学生はひところほど麻雀をしなくなって，学生街の雀荘は閑散としていると聞きます。しかし，本屋の軒先には何冊もの麻雀劇画雑誌が並んでいますし，相も変わらずゲーセンの片隅ではとめどなく新作麻雀ゲームが入るありさまで，飽きられて廃れていく気配は微塵もありません。ただ，パソコンゲームとしての麻雀に，ほかのジャンルのソフトほど新しいものを取り込んで成長していく力と魅力と度胸がなかったから。

そこに登場したのが，かの「ぎゅわんぶらあ自己中心派」です。麻雀ソフトといえば思考ルーチンの強化と高速化をまず考えるのが普通であった時代に，娯楽性を追求したソフトを見事に完成させた，ゲームアーツはさすがです。ぎゅわん自己の成功によって，にわかに麻雀ソフトが活気づいてきました。そうしてここに，麻雀ソフトは娯楽派と本格派に分断されたのです。

今回遊んだ3本はすべて娯楽派の代表作。X68000用の「麻雀狂時代SPECIAL」，X1 turbo専用の「今夜も朝までPOWERFULまあじゃん」，X1シリーズ1ドライブでもOKの「まじゃべんちゃー・ねぎ麻雀」。いずれもかる〜いナンパな麻雀です。

昔から，飲む・打つ・買うと申しまして，飲むは酒，打つは博打，買うは女なんですが，本能的快楽3要素があります。これらに頼らないと楽しめないというのは困った日本人ですが，なかなか男の本能的な部分を突いていて一概に否定できないところがつらいところです。

麻雀はいうまでもなく“打つ”のギャンブル。ギャンブルの醍醐味は人事を尽くして天命を待つ緊張感にあります。麻雀くらい複雑なゲームになってきますと，どこまでが人事で，どこまでが天命か判断できなくなり「ええいっ！　ままよ」と目をつぶって勝ったり，完璧な集中力と読みを持ってしても負けたり（もちろん最終的には後者のほうが強いはずだ）と奥深い勝負の世界が繰り広げられるのです。

しかし，それは相手が生の人間の場合。コンピュータ相手ですと，お金も人間関係もなにも関係ないので，つい適当に打ったり，コンピュータ側の3人が速く打つのでそれにつられて捨て牌をまったく読まなくなったり，運だけに頼ったりと投げやりな麻雀に手を染めてしまいます。おかげで麻雀の腕がかえって落ちてしまう百害あって一利なし。そこで，いかにユーザーに「負けてたまるか」心を起こさせるかがポイントとなってきました。シューティングやRPGなどほかのジャンルで培われたアメとムチ，プレイヤーを先へ先へといざなうノウハウが必要となってきたのです。

では，ということでゲーセン麻雀ゲームのポイントである“女の裸”とストーリー性を持ち出したのです。麻雀劇画に女はつきものですからね（3要素のうちの“買う”ですな)。勝ち続けないと女の子の裸が見えないという，ある意味ではたいへん卑怯な手を使ってユーザーに緊張感を強いること

となったのです。たかがデッサンの狂った女の裸なぞ……，と思う人もありましょう。そこは哀しい男の性，なのですね。

　では，麻雀ゲーム３本立てをどうぞ。

とりあえず，始めてみる

　まずは「麻雀狂時代SPECIAL」。今回遊んだのはX68000用だが，X1turbo用はかなり前に出ていたりする（と，いきなり文体が変わってしまった。気持ち悪いやつ)。違うところは操作性と絵と速さくらいのもので，ほとんど一緒だね。いきなり江戸時代風の絵とビバルディの四季より“春”には笑ったけど。

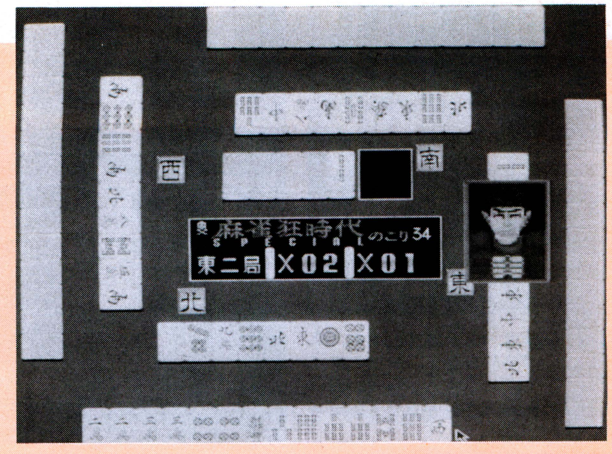
　X1turbo版と同様に２人対戦トーナメントモードと４人卓囲みモードの２部構成。秀逸なのはさすがX68000という牌の質感であって，これは４人卓囲みモードでより発揮される。従来の４人モード麻雀ゲームのように上から４人横に牌を並べている（そういえばぎゅわん自己もそうだね）んではない，きちんと四角く囲んでいるのだから思わず画面を上に向けて遊ばにゃあ，と思う次第だ。

　対して，２人対戦モードでは例によって対面式の勝負の世界。麻雀劇画でありがちな，主人公と敵役がバトルを繰り広げている間，残りの２人がただの座敷童，あるいは「異能ただおる」(知っている人，いますか？）と化してしまい哀れをさそうといった悲劇はないのである。でも，実際に２人で麻雀をすると，牌を積むのがたいへん難しいんだよね。

　普通の４人モードもいいが，今回は徹底的に娯楽性を高めるための付加価値を追求すべく２人で対戦した。２人対戦だとトーナメントモード，好きな相手を選べない。「ぼ，ぼく，あの女の子がいい……」とのたまっても「ふっふふ。君は世間をなめている」と，あしらわれるのがオチだよん。

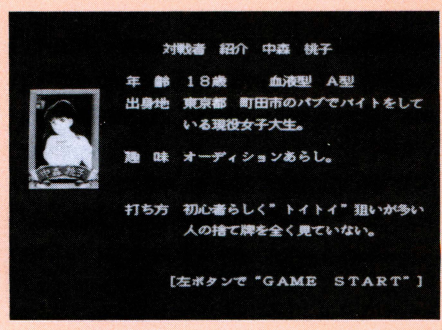
　続いてX1turboでは「今夜も朝までPOWERFUL まあじゃん」が起動する。あのスーパー春望とうってぃぽこのデービーソフトである。こちらは麻雀狂時代SPECIAL なんて目ではないモードの多さ，お好きな麻雀で遊んでくだせぇと，よりどりみどりつかみ取りである。メニューに並ぶはノーマル麻雀，エキサイト麻雀，さすらい麻雀，ぽこ麻雀の４種類。ノーマル麻雀なんてこのさいどうでもいいとして，選ぶはひたすら女の子を脱がせることが生き甲斐のエキサイト麻雀か，日本を南から北まで旅をして歩くさすらい麻雀か。こちらも相手を選べないさすらい麻雀を選んだ。さすらい麻雀では女の子の裸が出てこないのがさみしい，なんてことはいわない。私は聖人君子，正義と真実の荻窪圭である。





このゲームは長考していると，ウィンドウがポコポコ開いてせかされる。対戦相手は左下に登場の個性的な方々。それにしても中森桃子嬢の町田市のパブでアルバイトというプロフィールは笑える（右下）





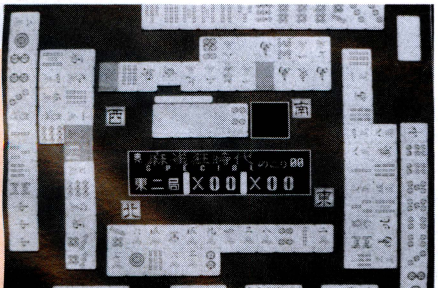
　X1turboがガーガーいっている間，その隣のX1twinでは「まじゃべんちゃー・ねぎ麻雀」である。これはもうテクノポリスソフトであるからして，あなたの予想どおり，女の子がいないわけがない。どうやらストーリーには関係ないところで勝手に女の子が脱いでいってくれるらしいが，女の子がどうなろうと関係なくゲームが進むところが「へっへ。見たいのなら見せてやるが本当の敵はそこにはいないんだぜ」的なアイロニーがあっていかにもテクポリらしい。

　流行なのか，突然X1が「マジャベンチャー，ネギマージャン」と喋ったのもテクポリらしい。スタート時に主人公のタマネギ(だと思う)が友だちの家へ遊びに行くのだが，その絵をラインとペイントで描くところがいかにも初期AVGのパロディしていてテクポリらしい（ほかの絵はちゃんと描いているよ)。１ドライブで遊んだので入れ換えがめんどくさいのもテクポリらしい（んなことはないか)。いきなり出てきた女の子が年端もいかぬしっぽの生えたガキであるところもテクポリらしい。こんな絵を「かぁいい」という身内の（で）氏はどうやらオタクらしい。まったく困ったものだ。私にはただ不気味なだけである。

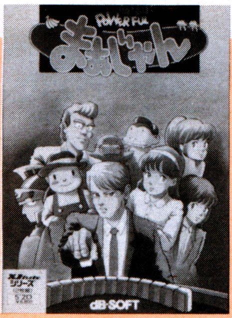
　おっと，X68000では第１回戦の対戦相手，女子大生で18歳で趣味がオーディション荒らしの中森桃子が退屈してマイク片手に踊っている。待ちなさい。すぐ相手してあげるから。うーん。X1シリーズのグラフィックのあとだとさすがにキレイだ。とりあえず，軽く“くいたんドラ３”の満貫でも狙うか。マウスは楽だ。鳴けるときはぺこっとウィンドウが開いて，いらないときはキャンセルして，ポコっと牌を捨てて，キーボードのように行き過ぎて隣の牌を切ってしまったとか操作を忘れて（私の記憶力はワンボードマイコン並みなのだ）リーチの度にマニュアルを開かなくともいい。

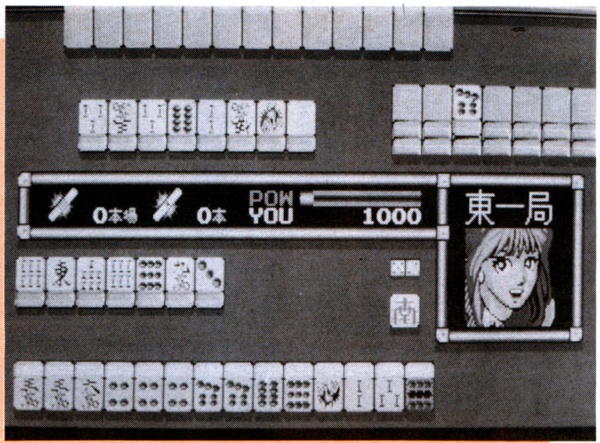
　それにしても，ずいぶんゲーセン版とは違うなあ。ゲーセン版は女の子３人だけだったしなあ。まあ，ゲーセンの麻雀もいまや麻雀狂時代なんて時代遅れの錆びた店にしかないし。いまは麻雀学園の時代だもん。

　とりあえず，くいたんドラ３でロン。お

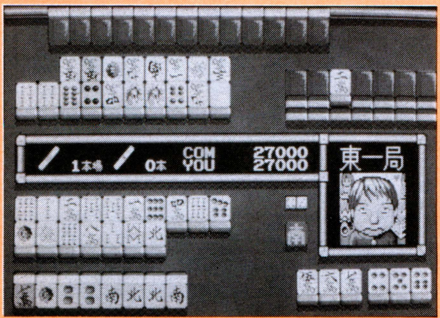


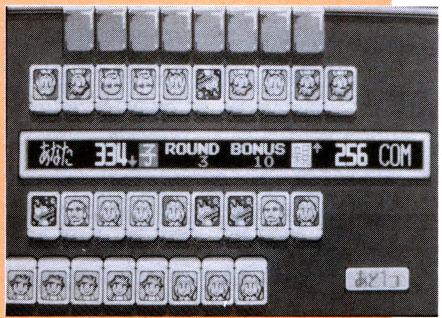
テンパってたのに残念でした





上からエキサイト麻雀，さすらい麻雀(左)，ぽこ麻雀(右)。さすらい麻雀の対戦相手は，誰が見ても陰険そのもの





いおい，バスタオルを巻いた桃子ちゃん。そんだけしか見せてくれないの？　ちょっとずっこい（ちなみに，ずるいっていう意味ね）よ。このモードだと２回上がれば終わっちゃうから，次は役満でも上がんないと全部見えないじゃない。ずっこいよなあ。ええい！　結局２回目も跳ね満どまりではないか。覚えていろよ。

気分転換に第２会場のさすらい麻雀へと移動。まず名前やら性別やら血液型やら年齢やら入れる。でも，性別“１”血液型“す”なんて入れても怒らないからいい加減なものだ。名前は最近気に入っている“ぬめば”とした。

モノクロの，こいつ，俺を笑わせようとしてるな!!　的のオープニングでいきなり沖縄の変態オタクと対戦。３万円持って行って，まず場所代3,000円だ。こんなやつさっさと片をつけるぞ，と。操作が全部テンキーでできるのが嬉しいね。

さて，これはタンヤオ狙いにいくか，三色も付きそうだ……なんだ？　考えても無駄だと。うるさい！　しかもオタク独特の癖をよく掴んだ丁寧語で攻めてくるではないか。この野郎。俺のマニア嫌いを知っての狼籍か。許せん!!　天に代わって成敗してくれる。えいや！　メンタンピンドラ１リーチ！

ふっふ……ん？なに５千点リーチだぁ？なんだそれは。マニュアルを見る。おお，５千点棒でリーチをかけるとウーハン（五翻）にもなるのか。ずっこいやつ。次からはそれでいこう。１万点リーチだと十翻だから，ほかに五翻あれば数え役満ではないか。なんて恐ろしい地方ルールだ。こうなりゃ容赦はしないぜ。上がって上がって上がりまくってやる。それ，１万点リーチにドラ２のみだ。タンピン１万点リーチだ。

と，勝ち続けたはいいがいつまでたっても終わらない。相手はマイナス10万点だ。そろそろ普通ならギブアップだ。マニュアルを読んでみる。

げげっ。こっちがギブアップ(SHIFT＋BREAK)するか，半荘終了までやらされるのか。南４局まで８回も，しかも親のときに連荘なんかすると永遠に，相手側が上がるまで局が進まない。そんなのなしだぁ。ひどいひどいひどい。ほら，敵のオタクが１万点リーチで上がってしまった。勝ち続けるより，相手には安く上がらせてこちらはでっかく上がるほうがむずいんだぞ。俺は怒った。こんなゲーム見捨てて第３会場のねぎ麻雀だ。

ねぎ麻雀ではいきなり，「がんばって」との声に送られてNAO KUNとの対戦だ。こいつもテンキーとスペースキーのみで操作可能だ。対戦相手の顔は見えないが，その代わり画面の３分の１ほどの面積を，ようわからん変な女の子（ロリコン雑誌やコミケで売っているいかがわしい同人誌に出てきそうなタイプ）が占めている。畜生。俺がこの手の絵が大嫌いなのを知っていて描いたな。許さない。しかも相手は時折「サンダーストーム」などとわけのわからん言葉を発する。まあ，３回闘って持ち点の多いほうが勝ちという短期決戦だから許そう。ひとり目はあっさりかわした。次はタンヤオのたっちゃんだ。その前に，第１会場のX68000では２回戦の相手，M. カトーがキセルを吸って暇を潰している。相手をしてやるか。

M. カトーが引き連れているのはゲーセン版でもお馴染みのでかいグラスに入った謎の美女だ。高い手で上がるとちっちゃい忍者のできそこないみたいなヤツがストローで半分くらいジュースを飲んでくれるのでサービスがよい。でも，ジュース(だよね)が減って直接見えてしまうより，ジュースに隠れたシルエットのほうがヒワイだという意見もある。

さっさと謎の美女のヌードを拝んで，再びさすらい麻雀に挑戦だ。

さて，第２会場。じっと我慢の子に徹し，なんとか長崎へと行くことができたが，「このやろてめ～，かんがえてねーできれったら」などといじめられて，体力と精神力はすでにリポビタンＤでも復活できぬほどに疲労した。ここはBGMでも変えてみよう。今夜も朝まで～麻雀（長い名前は嫌い）ではなんと静寂を含めて６曲までBGMを選べるのだ。歌謡曲，演歌，クラシック調からロック，ディスコまで。私は精神統一のため，静寂を選ぶ。が，長崎でぽろりと負けてしまった。なんて疲れるゲームだ。次は女の子と対戦するエキサイト麻雀で頭を休めようっと。

さて，第３会場のねぎ麻雀である。こちらも２回戦は楽勝。勢いにまかせて林くんと対戦だ。ここで勝つと第２部へと行ける。慎重に慎重に……あれ，なんだこいつ。少牌してやんの。ひえー。麻雀ソフトで少牌なんてどうやったらできるんだー。なんて画期的なソフトだぁ。おいらぶったまげっ



ちょっと上がったくらいでいやなやつ

たの仏陀。うーん。変なやつ。

第１会場では３回戦のα・ポックだ。なんと趣味はスター・ウォーズをけなすこと。うんうん，けなしていいよ，僕も手伝ってあげる。でもスター・トレックもけなすからね。ふっふ。引き連れるご婦人はポックの妻で夫と同様耳がとんがっていたりするが，顔が奇麗なら耳が尖っていようが長かろうが３つあろうが，細かいことは気にしない。それが男だ。

リポDパワー炸裂！

この恐怖の麻雀ソフト３本立て同時上映も，栄養ドリンクパワーでなんとか乗り切った。畜生，ナチュラルドラッグ以外には手を出さないと誓ったのに。

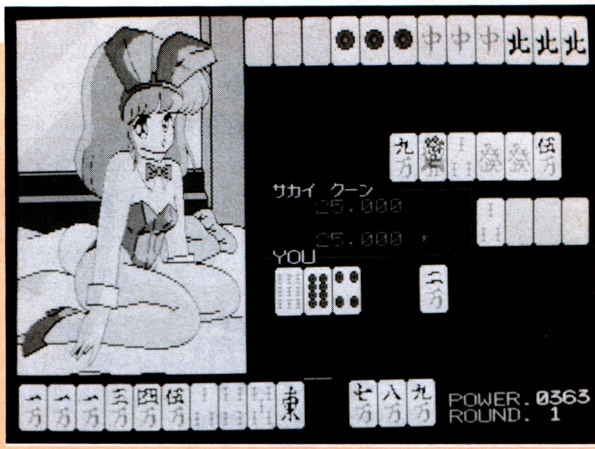
結果。第１会場では見事優勝し，画像取り込みの洋風の部屋のベッドの上でけだるそうにしている美少女を拝むことができた。やはりＢ級ゲームは面白い，というより普通のＢ級ゲームを面白くしてしまうX68000は凄いのだ。

第２会場では気を取り直し，エキサイト麻雀で楽しんだ。17歳の橋本奈緒美嬢より，21歳の高橋香織女史のほうが女狐してて面白かったね。許せないのはみんな気が短いこと。やれ「かんがえないで」だの「はやくすてれば」だのせかすものだから，つい間違えて捨てちゃって困ってしまってわんわんわわん，になってしまった。香織ちゃんは「か」，奈緒美ちゃんは「は」という感じでダイイングメッセージを残してくれたんだけどあれはなんだったんだろう。結構謎だったりする。全員に勝ってみないとわかんないだろうなあ。きっと，メッセージをつなげた名前でプレーするといいことでもあるのだろう。

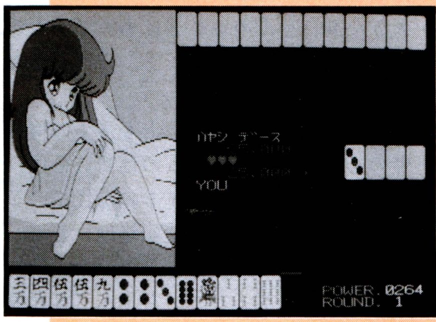
実をいうとおまけで付いてきた，うっでぃぽこのキャラクターが総出演する，ぽこ麻雀がいちばん気楽で簡単で楽しかったりするんだな，これが。もう少しぽこ麻雀に力を入れてもよかった気がする。

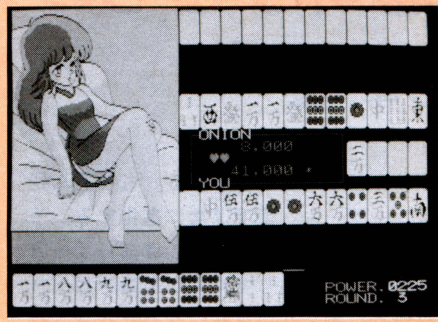
第３会場のねぎ麻雀は困ったことに第２部からイカサマができるのだ。上がると点に応じてパワーが貰えるんだけど，そのパワーを使って積み込みやらラストチャンスやらがある。役満積み込みで役満テンパイ(聴牌)だったりするんだけど，たくさんパワーを使う割にはそのあとのツモが悪くて敵さんに上がられると腹が立って健康によくないんだよね。だいたい，コンピュータがイカサマをすると怒る癖に自分はやりたいという，人間心理の襞を突いたいやらしさがあるね。テクポリにも困ったもんだ。





これがあの筋の方々が推薦していると噂の“うさぎちゃん”たち。それにしても，左下の写真の女の子の頭部はデカすぎる





きっと，ねぎ麻雀は相手もイカサマやってるぞ。

では，ついでだから，一部の読者が気にしているだろう女の子の評価でもしておこうか。

まず，麻雀狂時代SPECIALはさすがにX68000だけあって奇麗だしバラエティに富んでいるんだけれど，若いロリロリした娘がいないのでその方面が好みの方はあまり好かないだろうね。僕はたまにデッサンが狂っていても（そういえばこの手の女の子はたいてい人間業ではないプロポーションをしているもんだ），幽霊がヌードになったり宇宙人の美女がシャワーを浴びてたりするほうが楽しいけど。それにしても，幽霊のヌードが見られるという発想は，とっても人知を越えている。

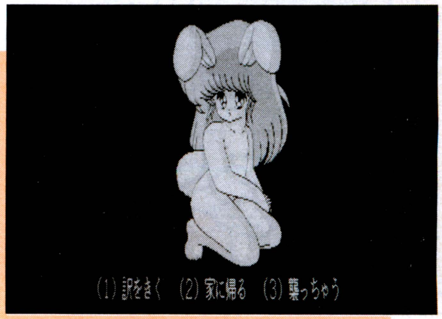
今夜も朝までPOWERFULまあじゃんの女の子は，今回の３本のなかではいちばんまとまっているね。７人いて16歳から23歳までさまざまだし。捨て台詞を吐きながら脱いだりして。これだけいれば，まあひとりくらいはポイントの高い子が誰にとっても存在するでしょうね。

ひどいのがねぎ麻雀ね。みんな胸なんてないくせにブラジャーは付けてるし，足はサリーちゃんだし。こんなのがいいという人の気がしれない。思わず画面の左上に(絵の出るところ）紙でも貼って見えないようにして遊ばないと，いろいろな意味で痛々しくてつらいものがある。こういった絵の好きな人，ごめんね。

ぐったりした私

さすがに麻雀ソフトばかり一気にやると疲れる。おかげで友だちと雀荘へ行ったら何年か振りで大負けしてしまった。麻雀ゲームはやりすぎると，なんにもしないより実戦の勘が鈍っていけないね。ほどほどにしましょう。

今度は，「雀豪１」や「悟空」みたいな本格的な渋い麻雀ソフトをしたい（結局，懲りていない)。雀豪１は賢くて，プレイ記録がディスクにいつまでも残るので面白そう(個人ごとのデータの持ち込みができるようになればもっといい)。誰はタンヤオが多くて，誰はトイトイが多いなんて癖がわかっていい。早くX68000版でも出ないかなあ。うーん。



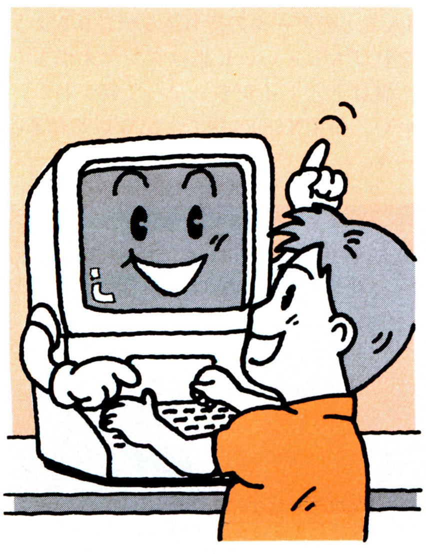
３番目のコマンドはいったいなんぢゃこりゃ

よりよいソフトウェア環境のために（最終回）

理想の環境が意味するもの

Tama Yutaka
多摩　豊

環境にはすべてが含まれる。もちろん我我ユーザーも。環境は常に進化する。それを理想の方向へと導くのはユーザー自身である。そうして，やがてコンピュータが目に見えない存在になったとき，我我が考えるべき問題も一歩進んだものになるだろう。



いよいよ最終回である。そこで今回はこの1年間に書いてきたことをまとめる意味も含めて，僕の理想とする環境，いってみればサイエンスファンタジーの世界をお届けすることにしよう。

すべての初めに

理想の環境すべてを支配する鍵，それは“子供”である。

環境については，まず対象となる人間を考えなければいけない。そして僕が理想とする環境は，すべての人がその利益を享受できるものなのである。“すべての人”といった場合，もちろん技術者やビジネスマンが大きな割合を占めることは事実であるが，忘れてはならないのが“子供”の存在である。

現代社会において，もっとも多くの情報処理の必要に迫られているのは若い世代，とくに子供である。成長するのが仕事の子供たちは，1日の大半を情報処理に費やしている。我々が大人面をできるのも，子供のころからさまざまなことを学んできた成果なのである。

世の中がより高度情報化社会へと変貌し，世間に出るまでに覚えなければならないことが増えていくのであれば，こういった“過酷な試練”に立ち向かわなければならない子供たちこそ，もっとも情報処理機械が必要だといえるのではないだろうか？　それゆえ，これから書いていく理想の環境は，常に“子供”を念頭において考えなければならないのである。

ハードウェア的な問題

理想の環境に必要なハードウェアを想定するなら，それは結局のところ“ダイナブック”に行き着くことになる。

ノートの大きさでコンピュータの機能をすべて持ち合わせているもの。ダイナブックのハードウェア的な要素をひと言で表すとこういうことになる。ラップトップコンピュータはこれに非常に近い存在であり，ハードウェア的な部分だけをとってみると，もうこれの完成は目前ということができる。しかし理想の環境を追うからには，当然のことながらこれに満足するわけにはいかない。

理想とするハードウェアは，軽くなければいけない。つまり，小学生でも手軽に持ち運べる程度で，当然のことながら厚みもできる限り薄くなければいけないのである。

現状のラップトップコンピュータの中で，重量や厚みを軽減する際の問題点は，バッテリ，ディスプレイ，ディスクドライブの3要素である。これらのうち現在の技術で利用できるものはICカードによる外部記憶装置の標準化程度であろう。これの採用により，ハードディスクやフロッピーディスクドライブの厚みや重さは軽減できる。

さて，問題となるのはディスプレイである。もともとコンピュータ自身は思ったほど電力を消費するものではない。電卓などは水銀電池でも駆動するのであるから，極言すれば，コンピュータを乾電池で動かしたっていいだろう（もちろん，現状では実用にならないが）。

ところがディスプレイがつくと話は全然違ってくる。子供が持ち運べることを目標とするハードウェアに，CRTディスプレイをつけることを考える人はいないだろうが，現在のところ液晶カラーディスプレイでもかなり重いし，なんといっても電力を食う。結局，軽量薄型低消費電力のカラーディスプレイの登場が，すべてを可能にする鍵なのかもしれない。もちろんこれに代わるなんらかの方法があるかもしれないが。

続いて周辺機器の問題も考えてみよう。

まず入出力関係はディスプレイとスピーカ，キーボードとトラックボールとICカードリーダにモデムあたりが必要最低限ということになるであろう。この中でちょっと奇異に思えるかもしれないのがトラックボールである。現状ではマウスが全盛であるが，あれは操作するのに一定の平面を必要とする。持ち運びができるということは，またどんな状況でも使えるものでなければならないということでもある。ところが，電車や車の中など平面が確保できない場所では，ポインティングデバイス（要するにカーソルを移動させる機器）としてマウスを使うのは難しくなる。よってそれに代わる装置としては，ジョイスティックとライトペンとトラックボールなどが考えられる。どれにもそれなりの長所短所はあるが，まったくの独断からここではトラックボールを採用することにしたい。

また，プリンタはやはり外部機器とするほうが現実的であろう。夢物語として超小型軽量低消費電力高速レーザープリンタの出現を想定してもいいが，どちらにしろ今回の話からはわき道にそれるのでとりあえず置いておくことにしよう。

操作の基本となるシステム

いわゆるOSは重要な要素である。この連載の初回にも書いたことだが，コンピュータの操作法はOSとマンマシンインタフェイスの両面から考えていかなければならな

い。この環境の対象は“子供”であるから，WYSWYG(What You See is What You Get)の思想に基づき，いわゆるコマンド入力などが必要ないインタフェイスと，机にノートや教科書をしまうのと同じレベルで把握できるOS概念が必要となるのであろう。

ダイナブック本体を机にたとえるとすると，さまざまな情報を管理するICカードはルーズリーフバインダにあたる。そして個個の情報（コンピュータ用語でいうところのファイル）が，このバインダの中の紙になるわけである。ちなみにこのたとえでいうと，ソフトウェアは情報を使う公式であり，また鉛筆や定規などの道具でもあるわけで，ソフトを入れておくものはさしずめ筆箱か鞄ということになるわけである。そういった観点からすると，データ保存用のカードとソフト供給用のカード，この2種類に対してそれぞれ別のカードリーダが必要になるのかもしれない。

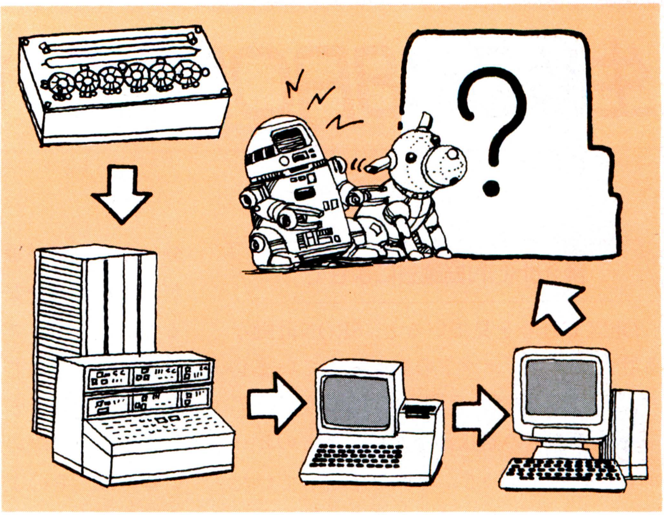
こうした概念を図式化して，それに基づく操作法を確立すれば，子供でも使えるものとなるはずである。そして，その操作方法はすべて一貫していなければならず，たとえばソフトウェアが異なると使い方が変わるというようなことがあってはならない。使っている者にとっては，自分がOSレベルにいるのかソフトの中にいるのかなどということすら認識できなくてもよいわけである。

データの問題

次にデータについて考えよう。

情報の交換（これはノートの貸し借りにたとえることができる）は単純にカードの交換でなされなければならない。これが実現するためには，どこのハードウェアを使っていようと，まずカードのフォーマットは同じものである必要がある。

次にデータ自身のフォーマットだが，「どういったソフトを使っても基本的に同じフォーマットでデータを保存する」などという方法論をとるのは誤りであろう。基本的なデータ（文字，画像，音声など）に関して，それぞれ基本フォーマットは定めておく必要があるが，ソフトごとにこのデータを操作する方法は異なっていてもよいと思われる。要は基本フォーマットに変換することができ，基本フォーマットから変換することができる機能を，すべてのソフトが備えていればいいわけである。この操作方法が十分に簡単であれば，子供でもこれを利用することはできる（「基本の形に戻す」という概念は，積み木を箱から取り出し，それをさまざまな形に作り上げ，それをまた元の箱にしまうというような例によって，子供にでも容易に理解できるであろう）。



たとえばここにソフトAで作ったファイルがある。このファイルをソフトBで直接読むことはできないが，ソフトAはファイルを基本フォーマットに変換でき，ソフトBは基本フォーマットのファイルを自分で使えるフォーマットに変換することができる。よってソフトAのファイルはソフトBでも使えるわけだ（ちょっとばかり面倒臭いが）。

大事なのはこの基本フォーマットが優れたフォーマットで，どんなソフトを開発するときでも，必ずこれとの相互変換が可能な環境が作られることである。これさえ確保されれば，結局のところデータは常に互換性を保持することになろう。実際問題としては，このデータ変換をより容易にするようなソフトが出現することは間違いないが。

ソフトウェア

この環境に対してソフトウェアはどうあるべきか？　まず，基本的に提供されるべきソフトウェアは，ハードウェアと一体化しているべきであろう。

ワープロ，スプレッドシート，データベース，グラフィック，サウンド，通信。この6種類に関してはROMベースで存在していなければならない。それぞれのソフトウェアは非常に頻繁に使われるものであり，またあまりさまざまな種類があっては使う側に混乱をきたす恐れがあるからである。

こういったソフトウェアは，その使い方が子供にでも理解できる程度に（たとえばスプレッドシートなどは小遣い帳なみに）簡単であれば，即座に世間に受け入れられ，利用されるだろう。反面，その程度ではより高度な利用には機能不足となるかもしれない。そこでカードベースで供給されるソフトウェアが登場するわけである。ただしこれも，ROMベースで供給されているソフトの機能を強化するようなものとなるのである。

たとえばスプレッドシートを強化する定形表ソフト，ワープロを強化する辞書ソフトや文形ソフト，グラフィックソフトを強化するツールソフトなどだ。これらが元のソフトの機能を強化することにより，ユーザーは新しいソフトウェアと格闘することなしに，より高機能で用途にあったものを即座に手に入れることができるようになるのである（さらに，これですべての要求に応えることができれば，データの互換という問題も考慮する必要がなくなる）。

よりよい環境とは？

こういった環境の実現は，社会に大きな変動をもたらすかもしれない。たとえば通信も大きく変わり，ファクシミリや郵便の代わりに直接データ転送するようになるであろう。大型コンピュータとの接続もより容易となり，大規模ネットワークサービスが新しいマスメディアとして開花する。そしてダイナブック抜きにしては，生活自体が成り立たないという状況も訪れるかもしれない。

これは情報化社会の理想像の一端である。決して現在のパソコン環境を取り巻くハードウェア，ソフトウェアの非互換，とどまるところを知らない高速化，多機能化の嵐，そしてより専門的で複雑となっていく操作性などを推し進めた結果として得られるものではない。

よりよいソフトウェア環境を追い求める厳しい批判の目，それが存在して初めてソフトウェアの変革がなされ，ハードウェアの進化も可能となる。そしてこれこそが，次なる理想的な環境を生み出していくための鍵なのである。

パーソナルコンピュータ。それはいまだに“環境”と呼べるような状況を作り出してはいない。

生ぬるい8RONならいらない!

僕を8RON協議会にまぜて

Oh!X 6月号を見ていると，8RON計画なる壮大なプロジェクトの話が載っていました。まったく喜ばしい限りです。こういうアーキテクチャの話は汗をかいたあとのビールぐらい好きなので，今回は予定していたテーマを急遽変更してこのことに関していろいろ書きます。8RON計画を陰ながら応援していこうと思っているのです。

ですから，6月号を見ながら読んでくださいね。最初に，祝氏がなぜ8ビットか？ということについて書いています。8ビットプロセッサという言葉の定義は，プロセッサ内の演算ユニット部で扱うデータの幅が何ビットであるかということですから，単純に考えれば，このビット数が大きければ大きいほど1回に扱えるデータ量が大きくなるので，スピードが速くなるわけです。でもあえて8ビットとこだわる気持ちは次のようなものでしょう。

1) 16ビット，32ビットとなるにつれていろいろ機能が豊富になってきてしまい，本来のせっかくのプロセッサの機能から目がそらされがちになる。

2) とにかく安いので気楽にプロセッサを買い込んでそれを使って好きなシステムを組めるし，コーヒーをその上にひっかけてしまってもすぐに立ち直ることができる。

祝氏がディスクの問題に言及している部分は，一見話が飛んでいるようにみえますが，これがなかなか渋いところをついているといえます。プロセッサの世界はなんとなく華やかなので目がいきがちですし，実際に研究の世界でも花形のほうに入ると思いますが，実際のマシンの性能を落としているのは，そのような一見地味で泥臭い部分である場合が案外多いのです。そういうわけで，このへんのことを由々しく思っていた自分としては，まったくそのとおりといいたいところです。

最後の村田氏の「たこあし君」は，現実的のようで非現実的で，でもやっぱり現実的で面白いものです。今ではもうプロセッサは中心に偉そうに座っている指揮者という考えでは対応できないような場面がよくあるのです。つまりちょっといいメモリや周辺機器などをつなごうとするとプロセッサよりずっと高くなったりするのです。やっぱり計算機の中でも，ものの値段というのは大きな影響があるわけでして，今ではもうプロセッサもメモリや配線やバスなどのようなシステム資源のひとつとして平等に扱うような態度を持たなくてはならないようです。

さて，いよいよ中森氏の「V8」チップ提言の迫力ある記事です。8個の特徴が書いてありますが，比較的当たり前のことと，とんでもないことが並べてあるところが特に素晴しいところです。究極の8ビットといいながら，「Z80完全互換」というのもちょっとみみっちくて可愛らしく思われます。

1)の豊富なレジスタセット，2)の命令の対称性，3)のメモリマッピングは，まあ一般的ですね。5)に書かれている物理アドレスを全部内蔵キャッシュに入れてヒット率100%にするというのはキャッシュの定義からいくとまるで変態ですが，まあそんなもんでしょう。

7)の自己書き換えサポートというのは思想的に好きでないとしかいいようがありません。百害あって一利か二利しかないような気がします。これについてはそれだけです。

残された4)，6)，8)はたいへん好きなところなので，ウリウリといきましょう。僕自身が考える究極構想の中でときおり関連させていくことにします。

ところで計算機のアーキテクチャというものは，フォン・ノイマン型の計算機が発明されて以来，多くの人がいろいろ研究に専念しているものの，実際に開発され，使われている計算機は，初期の計算機に比べて実はなぜかたいした進歩も遂げていないといえると思います。そのことは，現在日本でガンガンやろうとしているTRON計画でも，そのアーキテクチャはフォン・ノイマン型であるということをみても明らかでしょう。

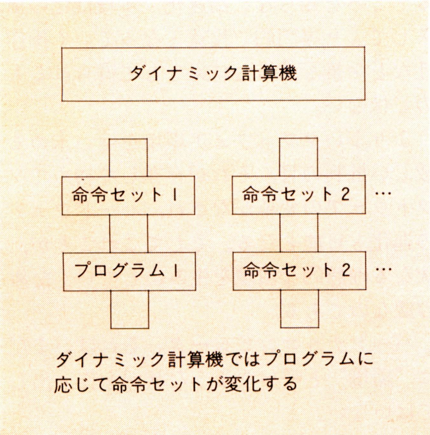
わずかに仮想記憶，パイプライン制御，キャッシュ，そしてRISCというのが小さな変革（というより技術的進歩）といえましょう。

いろいろ浮わついた計算機も考えられますから，本当はそちらのほうが面白いと思うのですが，8RON計画では，せっかく8ビットを見直そうということなので，従来の技術や流れをジャンプ台としてできるだけぶっ飛んだアーキテクチャをいろいろ提案したいと思います。

究極のダイナミック計算機

中森氏の提案4)の，ベンチマーク命令を用意し，ベンチマークテストプログラムを1命令でやってしまうというのは強烈な風刺も効いた痛快な提案です。一方，8)で述べられている，命令をユーザーが勝手に付け加えられるというのは比較的昔からあるマイクロプログラマブルプロセッサの考え方だと思います。最近のプロセッサは命令自体を直接ハードウェアで実現せずに，その下のレベルのマイクロプログラムをプロセッサ内に持って実現する方式が主流です。マイクロプログラマブルプロセッサでは外付けのROMに好きなマイクロプログラムを書いて自由に命令セットを設定できると

図1　ダイナミック計算機



いうものでした。

ところで4)と8)をいっしょに考えてひとひねりすると生まれるのがダイナミック計算機なのです。このマシンは，実行するプログラムを解析して，命令のシーケンスを調べ，最適になるように命令セットをマシン自身がダイナミックに決定するというものです。

それぞれの命令は，ミクロにみると計算機の中のいろいろな回路のゲートを開いたり，信号を作ったりしているのですが，命令と命令の間には独立した処理，つまり同時に実行できる部分が多数存在します。そういうふうにして命令をいろいろ融合すると，そのプログラムの実行にいちばん効率的な命令セットというものが実現するというわけなのです。

独立して実行できる2つの命令はひとつの命令にしてしまい並列に実行するというだけでも，性能的にはかなりよくなるでしょう。

プログラムの解析も，時間のかけ方によっていろいろな深さが考えられます。究極的には4)のように，プログラムを1個の命令だけで実行してしまうでしょう。しかし中森氏も指摘しているように，MIPS値(1秒間に実行できる命令数)がとんでもない値になってしまうのは，仕方のないことといえます。

究極のRISC

Sunワークステーションの最新シリーズがRISC型のオリジナルプロセッサを採用し，しかも高いMIPS値を実現したということは，RISC vs. CISC論争のひとつの決着を示しているといえるでしょう。

もちろん，製品レベルに至るまでにひとことでRISCといってもその定義の範囲はだんだんと広がっていきました。その意味ではRISC vs. CISC論争でCISC側に立っていた人たちにとっては，この論争自体すでに意味のないものとなっていると思われます（もともとCISCというのもRISC派が付けた言葉ですが)。

Reduced Instruction Set Computerという名のとおり，次々と複雑な命令が付け加えられていくことが結局は損になるのだということがそもそもの発想であり，そのために命令セットを縮小する，つまり命令数をとにかく少なくしたものが，忠実なRISCといえるでしょう。

究極のRISCは命令数がいくつになるのでしょうか？ 厳密に考えるとハードウェアに直接関係するところから始まり，計算機の定義というところにまでいくかもしれませんが，よく使われる命令は

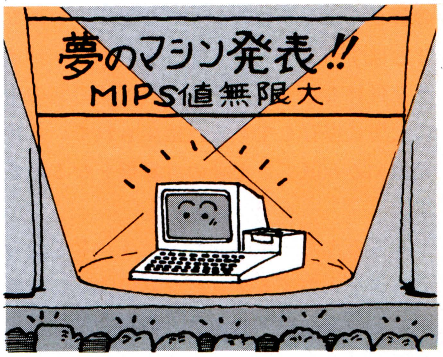
1) データの移動
2) データの演算
3) 制御の移動
4) 入出力命令

ですから，一応そこだけを考えればよいでしょう。

結論からいえば命令はひとつになるのです。要するに条件付きデータ移動命令だけでよいのです。それでは上に挙げた4つのうちの1番めしか実現されないように思われるかもしれませんが，まず3)の制御の移動(ジャンプやコールなど）はPC(プログラムカウンタ）へのデータの格納と同じですから含まれますし，4)の入出力命令に関してはI/O関係はメモリマップドとすればメモリへの読み書きと等価になります。

2)に関してはややうさんくさいと思われるかもしれませんが，これはアドレッシングに含ませてしまうのです。従来のマイクロプロセッサでも，アドレスレジスタを使ってアドレッシングを行うときにアドレスレジスタ間の加算は当然のことでした。ですからアドレスレジスタとデータレジスタの区別をなくして，アドレッシングの考えを拡張して演算も含ませるのは，一概に不自然とはいえないでしょう。

しかもその命令は前の演算の結果に基づくフラグが立っているときのみ行うので，種々の条件付き命令にも対応することができます。



究極のRISCではアドレッシングの持つ意味が重要になっておりそのために命令数を1個にすることができました。ところで命令数が1個ということは命令を構成するオペコードとオペランドのフィールドでオペコードはひとつであるということを意味します。ということはオペコード部をデコード(解読)する必要はなくなります。要するにオペコード部は不要なのです。したがって命令数はゼロともいえます。命令数がゼロの計算機，これでも計算機なのです。

無限大MIPS値をもつ夢の計算機

最近のコンパイラは定数の計算，たとえば，

x＝10＊45－27；

などという計算があるとコンパイルの時点でこれを

x＝423；

というコードに変えてしまうということは当たり前のようです。さらに

x＝factor(100)；

などというように，引数が定数である関数はあらかじめ計算してしまい，xに単にその結果の定数を代入するコードを出すというコンパイラも常識的になってきているそうです。

これを初めて知ったときはなぜかムショウに「反則っ！」という気持ちがしたものでした。でも今ではなんとも思わなくなってきました。

というわけで提案するのが，MIPS値が無限大という夢の計算機です。賢明な読者

の皆さんはもうおわかりでしょう，やれることはすべてやってしまうというコンパイラを付けるのです。そうすれば僕の家の中に大切に保存してある（捨てられている）元祖MZ-80Kでさえ夢の計算機となるのです。

もちろんキーボードやファイルなどからデータを入力するプログラムのように，解がひとつでないプログラムの場合は計算する部分はかなり残ってしまうでしょうが，そうでない場合には実行時間は厳密にゼロなのです。

実はここのところの話は今やっている研究の出発点ともいえる話なのですが，でもたぶん読者の多くの方は，「反則っ！」と思われているでしょうね。僕も同感です。

究極の高級言語マシン

計算機に関する多くの問題は，計算機のハードウェアとプログラムの意味的な隔たり，つまりセマンティックギャップにあるとマイヤーズはその有名な著書（参考文献2)で指摘しました。そのセマンティックギャップを埋めるための直接的なアプローチが高級言語マシンのアプローチです。高級言語というのはBASIC，PASCAL，C（これはそうでもないのだが，ここらへんの議論は次回でやるつもり）などのプログラミング言語や中間言語などであり，要するにアセンブリ言語よりはレベルが高いということをいっているのです。そして高級言語マシンはそのような言語を直接アーキテクチャでサポートしようというマシンです。

そのようなマシンでも究極といえそうなのが，直接実行型高級言語マシンと呼ばれるものです。これは，普通のプロセッサが命令をひとつ取ってくるような感じで，たとえば“if”文の“i”の文字をメモリから取ってきて処理を行っていくというたまげたマシンです。さすがに提案したChu氏らはまだ試作にまでは至っていないようですが，日本では数年前に筑波大学のグループが汎用言語型の試作機を完成させました。

残念ながら，速度的にはあまり期待できないと思います。なぜならば直接実行型マシンでは翻訳あるいは変換といったコンパイル的処理までを実行時にハードウェアにやらせているからだと僕は思っています。

このタイプのマシンは趣味ですので，そのうちオリジナルマシンのアーキテクチャを紹介できるでしょう。

究極のパイプライン計算機

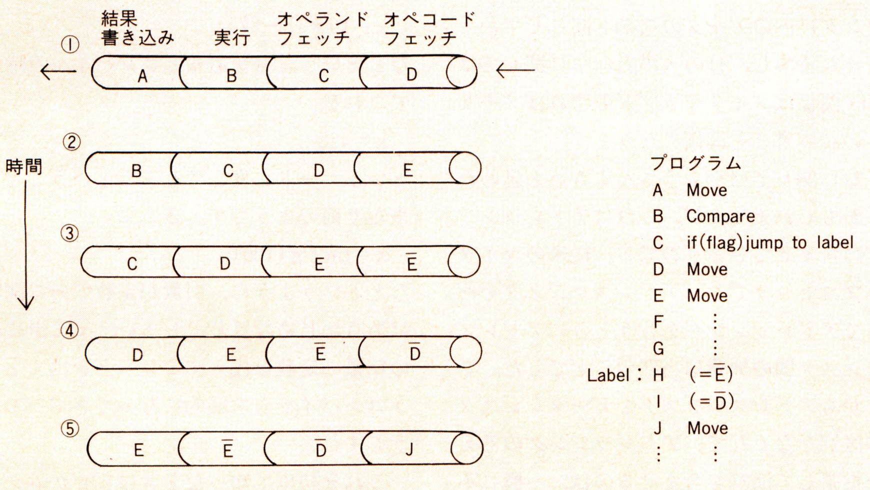
ある仕事をパイプライン的に行えばそのパイプラインの段数倍の速度が実現される，ということはもちろんプロセッサにも当てはまります。その場合，条件ジャンプのような命令があるとパイプラインが崩れるのがいやなので，条件が成り立つかどうかを予測したいのです。そこで，中森氏の6)に書かれている完全分岐予測機能が欲しくなるわけです。別に中森氏のユーモアに水を差すつもりはないのですが，これはまあキーボード入力によって処理を決めるような場合など，わざわざ例を出すまでもなく不可能なことです。

実行中においてこのような条件ジャンプが何度も出てくるのはループの部分です。つまりある条件が成り立てば次の命令にそのまま移動してループを抜け出るが，そうでないときはループの先頭にジャンプすることを繰り返す，などという処理を行うときです。

でも，パイプラインに読み込んでいくのはいつも先頭にジャンプするほうの命令列とする，というシンプルな方式が案外うまくいくのです。つまり，ループを抜け出るときの1回だけは失敗しますが，あとは全部うまくいくわけですからね。そういうわけで，後方に条件ジャンプするときはそっちにいくと見なしてパイプラインに入れてしまうとすれば，なんとかうまくいくと僕は思っています。それ以上がんばって推測したとしても，労力の割にたぶん数%速度が上がるといった程度の話ですから。

ところで究極のパイプライン計算機ではほんとうに100%パイプラインを崩すこと

図2　パイプラインにおける打ち消し命令の例



この図は4段のパイプライン計算機を示している。時間②において，Cを実行すると初めて条件ジャンプをすることがわかるが，ジャンプ先にジャンプしないとしてパイプラインに入れてしまったDとEを打ち消す命令，$\bar{D}$と$\bar{E}$が入れてあるので，パイプラインを無効にする処理が必要ではなくなる。

はないのです。かなり発想の転換を必要としますが，命令すべてにその操作の逆操作の命令を対応づけておきます。そして，分岐の起こるところではどちらか一方が起こるとして命令をパイプライン的に実行していき，もしそちらが成り立たなかった場合でも，あわてて読み込んだ部分をはき出して別の命令列の実行に移ったりせず，余分に実行してしまうような部分を打ち消すという命令列を実行するようにプログラムを作っておけばよいわけです。図1にこのようすを示します。

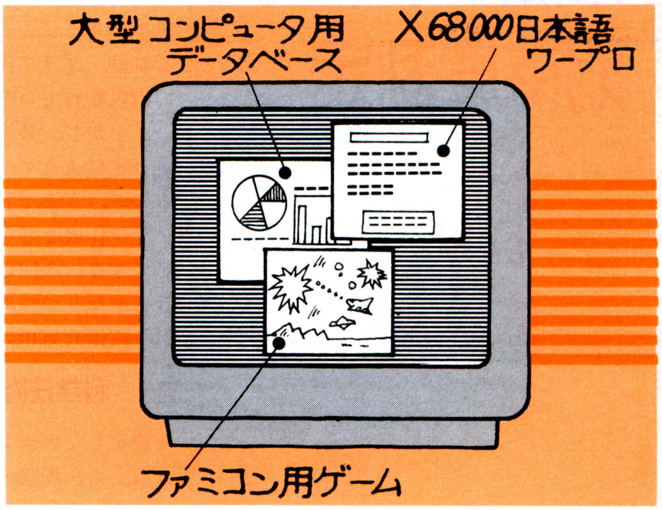
この方式は，VLIW計算機という参考文献3)に紹介されているマシンの中で使われているメカニズムに準じたものですが，かなり面白いと思います。VLIW計算機自体はALU（演算ユニット）を多数もたせた面白いアーキテクチャをもった計算機ですので，そちらも参考にしてください。

究極のウルトラコンパチ計算機

仮想メモリ，ダイナミックなメモリセットなどから始まり，仮想OS，上位コンパチ，PC-9801/IBM-PC対応パソコン，仮想マシンなどなど，一枚皮を被せてフレキシブルにしようとするアプローチはいろいろなレベルで見られます。

しかし残念なことにそのように仮想的にすればするほどスピード性能は落ちてしまうのです。32ビットだからこそ16ビットのエミュレートをしても実用になるということです。

一方，ユーザーの心理は自分のマシンがX68000にもなり，UNIXマシンにもなり，たまにはPC-98にもなるものがほしいということでしょう。Apple GS のように巧妙に2つのマシンのアーキテクチャをバスレベルで結合した例もありますが，ややもすると，別々に買ったほうがかえって安い，速度的にも機能的にも性能が落ちる，といったことになってしまうのです。そこでプライドを捨ててきれいでインテリアにもなるような箱を作るのです。さすがにただいろいろな製品をその中に突っ込めるようにするというのでは芸がないので，入出力だけは最高のものを用意しておきます。そうして種々のパソコンの入出力部分だけをうまくエレガントに取り換えて，このウルトラコンパチ計算機で統一的に扱えるようにすればよいのです。



入出力規格の統一的な扱いが成功した日にはもうどんな計算機が出現しても怖くありません。それを買ってきてこのマシンに入れて画面やスピーカやキーボードやマウスやディスクのところをアダプタにつけてもっと素敵なこのマシンのものに接続すればよいのです。

さらに通信機能をリッチにすれば，遠くにあるスーパーコンピュータもひとつのウィンドウとして，ファミコンでゲームをしているウィンドウの隣に開いて，ちょっと気象予測などをさせてみる，なんてこともできます。なかなか粋ですね。これこそ，究極の仮想マシン・ウルトラコンパチ計算機の真骨頂といえましょう。ちょっと場所をとるので押入れシェイプタイプをバリエーションとして用意したほうがいいかもしれません。

夢のマシンも夢でなくなる

今回は8RON計画に触発されて，急遽僕が考える究極的で，かつ比較的現実的な(そうともいえないか？）計算機を挙げてみましたが，命令セットを自分で決めてCPUを自分で作るということは，そんなにべらぼうなことではないのです。基本はデジタル回路ですし，頭で考えたことをそのまま回路で表現すれば，なんとか動くだけは動くというマシンはできると思います。もちろんここではVLSI化ということをいっているのではありません。

いちばん簡単なCPU（数ビットというおもちゃ）ならば，TTLシリーズとメモリとクロック発生器ぐらいでできてしまいます。またZ80程度の機能を実現するのでも，ビットスライスのALUとシーケンサを使ってやれば，特にハード屋さんでなくても（僕もハード屋さんではないのですが），それほど苦労しないでできてしまうのではないでしょうか。

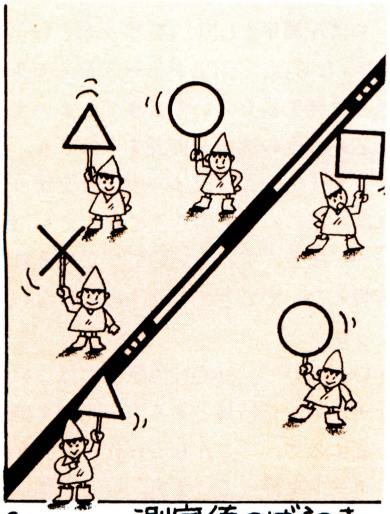
というわけで，8RON協議会ではこのようなチャレンジ精神のある方の出現を望んでいますので，「こんなもん作った」などのレポートを待っています，などと私は8RON協議会の名を勝手にかたってしまいます。

もちろんこの連載に関する意見や感想などを寄せてくださることも陰ながら待っているのですけど。来月は「ソフトウェア物理学」なる話をします，たぶん。

参考文献

1) 8RON協議会：8RON計画，Oh!X 6月号，pp. 88—95，1988。
2) G. J. マイヤーズ：コンピュータ・アーキテクチャの設計，共立出版。
3) 相磯，飯塚，坂村編：ダイナミックアーキテクチャ，共立出版。
4) 箱崎，山本：高級言語マシンの実際，産報出版。
5) 鈴木則久：研究者のテイスト，bit 5月号，pp. 35—40，共立出版，1988。

市販ソフトの期待度測定



Katsumoto Shin
勝本　信

科学技術計算用のグラフ作成支援ソフトが少ない。理工系の人々の多くは，プログラムは自分で書くものと思っており，市販ソフトの使用を潔しとしない傾向があるようだ。このため，彼らの多くが使用する最小二乗法によるデータ解析やGP-IBの機器制御，グラフ作成等に関するプログラムで市販されているものは数えるほどしかない。しかし，最近は表集計ソフトの多くが統合化によりグラフ作成をサポートしているので，うまく利用するとデータ解析からグラフ作成までを一貫して手早く行うことができる。今回は，表集計ソフトの標準的存在となりつつあるLotus1-2-3を取り上げ，科学技術計算用のグラフ作成を考えてみる。

科学技術計算用のグラフ

測定したデータを点としてプロットし，その上へ，理論から予想できる曲線を当てはめるというのが科学技術の分野で通常用いられるグラフである。たとえば，電流-電圧特性の測定では，素子に流れる電流をいろいろ変えて，どれだけ電圧がかかるかを測る。そうして得た電流，電圧の値をそれぞれX軸，Y軸にプロットする。素子が単純な抵抗の場合，データ点はオームの法則から定められる直線の近くに並ぶことが予想される。抵抗の値Rはできるだけ多くの点の近くを通るように引いた直線の傾きから求められる。直線の傾きを正確に決めるには最小二乗法を使えばよい。データ点のばらつきは測定の正確さの指標になる。電流の多いところで直線から外れる傾向があれば，たとえば発熱などで抵抗の値が変化しているのではないかと予想される。

いくつかの素子について同じように電流-電圧特性を測定し，ひとつのグラフに書き込むことも多い。この場合，素子それぞれを表すシンボルを変えてプロットする。測定の誤差も重要である。電圧計の表示が何桁か，最小桁の表示がふらついていないか，接続リード線の抵抗や接触抵抗はどの程度か，などを考慮して誤差が何%かを見積もり，プロットしたデータ点の上下に誤差の大きさを表す棒(エラーバー)を書く。

抵抗の電流-電圧特性や，バネの伸びと重りの重さのように，データ点が直線的に並ぶ場合は簡単だが，理想気体の体積Pと圧力Vの関係のように反比例するときはグラフは曲線になり，本当に反比例しているのかどうかわかりにくい。そこでY軸にVを，X軸にはPの逆数をプロットしてやると，データ点はきれいに直線上に並ぶので反比例からのずれなどを詳しく調べることができる。このように，測定データをそのままプロットするのではなく，関数（この場合は逆数という関数）をとってプロットすることが多い。

要求される条件

事務用グラフソフトを科学技術計算で使う場合はどのような点に問題があるのかを，以下にチェックしてみよう。まず，グラフ形式の種類であるが，棒グラフ，円グラフ，帯グラフなどは使用できない。

実のところ，実際に使用できるグラフの形式はXYという1種類しかない。これはXとYの値をそのまま横軸と縦軸にプロットするという単純な方式であり，散布図とも呼ばれている。もともと2つの現象の間に相関関係があるかどうかを調べるためのものだ。たとえば人間の身長をX，体重をYとして適当な集団についてプロットしてみれば，身長と体重の間に関係があるかどうかを調べられる。ここで我々に必要なのはX軸の値も指定できるということである。これ以外の形式では，X軸は多くの場合自動的に等間隔にとられてしまうので使用できない。

たとえば電流-電圧特性の測定の場合，電流の値が完全に等間隔である場合のみ，XY以外のグラフ形式もなんとか使用できる。しかし実際の測定では完全に等間隔ということはまずないだろうから，実質的にXY以外のグラフ形式は使えない。その場合XY形式をサポートしていないグラフソフトは致命的だ。たとえば，チャートupなどがそれで，いくらユーザーインタフェイスが素晴しくとも，肝心な機能が不足しているのではなんにもならない。

データの値そのままではなく，関数をとった値をプロットする機能が必要であると述べたが，その関数としては逆数，二乗，対数，指数などはもちろんのこと，ユーザーが定義した関数を使用できることが望ましい。というより不可欠であるといったほうが正しいだろう。関数をいろいろ変えてデータ点が直線に並ぶ関数を探す，という作業が，未知データを解析するときにまず行うことだからだ。

データの取り込みからグラフ作成までを手早く行えることも必要だ。測定を行いな

がら，データが少したまった段階でグラフにしてみるということがしばしばある。そんなとき，いちいち別のソフトを立ち上げるなどということはできれば避けたい。この点でMultiplan＋MS-Chartの組み合わせは残念なことに使いにくいものになってしまっている。

Lotus1-2-3をチェックする

使えるグラフ形式は上で述べたようにXYのみであるが，マニュアルや解説書を見ても2つの現象の相関を調べるためのものだとだけ書かれており，X軸の値を等間隔でなく自由に取れるという重要な点には触れられていない。せめてひと言でも書いておいてほしかった。

データファイルはテキストファイルである限り，どんなものでも読み込めるといってよいだろう。データの区切りもコンマやスペースなど適当でよい。一太郎で打ち込んだデータでも，アスキーセーブされたBASICプログラムのDATA文でもテキストファイルならなんでも読み込んでしまう。その他，SYLK形式はもちろんのこと，dBASE-II/IIIのデータファイルやK3形式なども付属の変換ユーティリティ経由で読み込むことができる。

最小二乗法を使ってデータに直線を当てはめるコマンドを内蔵しているところなどは，さすが表集計ソフトと言えるのであるが，パラメータの誤差の計算は直線の傾きのみであり，切片の誤差は求められない。しかし，なにしろ表集計ソフトであるから自分で式を打ち込んで計算させればよい。このあたりはグラフ作成専用ソフトにはない強みであろう。

関数プロットのやり方についてだが，適当な関数をとって，データ点が直線上に並ぶようにプロットすると見やすくなることは既に述べた。実例を示すため，データをXとYの組合せとしよう。まず，XとYが反比例の場合は横軸にX，縦軸に1/Yをプロットすればよい。データYがB行に1列から縦に入っているとすると，C行1列に，1/B1と入力してそれを下方にコピーすればよい。Lotusが自動的に式を1/B2，1/B3，1/B4……と変換してくれる。

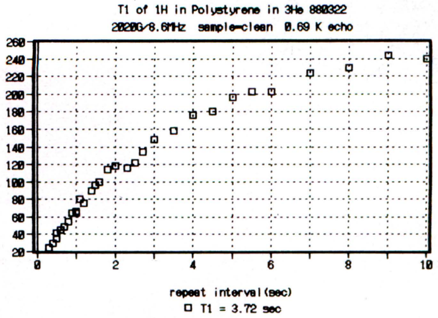
データYがXの指数関数になっている場合は対数をとってプロットする。いまの1/B1の代わりに@LOG(B1)と入力すればよい。ここで対数関数LOGの前に@をつけたのは，単にLotus1-2-3での規則に過ぎない。ではYがXのn乗になっていたらどうするか。この場合はXとYの両方の対数をとってプロットする。もともとの関係が$Y=AX^n$であるから，両辺の対数をとれば，LOG(Y)＝LOG(A)＋n・LOG(X)となる。こうすればLOG(Y)とLOG(X)が比例関係にあるのは明らかだろう。この場合nの値を知らなくとも，対数をとってプロットするだけで直線の傾きからnがわかる。

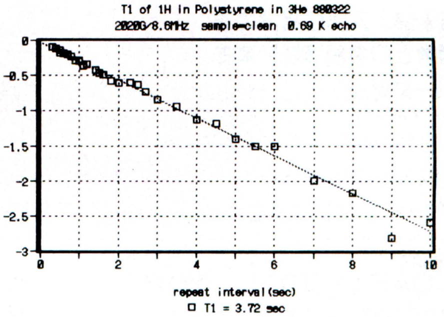
もっと複雑な例を挙げてみよう。Xの1乗と3乗の組合せ$Y=AX+BX^3$では，両辺をXで割れば$Y/X=A+BX^2$となるので，Y/XとX^2をプロットする。ローレンツ曲線と呼ばれる$Y=A/(X^2+B^2)$の関係では，1/YとX^2でよい。正規分布を表すガウス曲線$Y=EXP(-X^2/A^2)$では，対数をとってみればすぐわかるように，LOG(Y)とX^2でプロットすればよい。データ点が直線の近くにきれいに並ぶはずである。ここで挙げた例は，いずれも関数をとればデータ点の並びを直線に変換できるというものであるが，さらに複雑なもの，たとえば$Y=A(1-e^{-X/C})$などはAの値を知らないと変換できないため，最小二乗法で前もってAを求めておく必要がある。それからLOG(A－Y)とXでプロットするのである。

最小二乗法で求めた曲線を描く方法を説明しよう。曲線上のいくつかの点の座標を入力すればよいのであるが，もちろん実際には横軸の値を適当な間隔で入力し，縦軸の値は式を入力して，計算はLotusに行わせる。測定データ点が多いときはデータ点と曲線とで横軸の値を共通に選んで構わないが，データ点が少ないと，曲線が折れ線になってしまう。曲線表示用の点を十分な量だけとって，測定データ点とソートしておけばよい。

問題点

グラフの下書き用にはほぼ満足のいくLotus1-2-3であるが，清書用にはまだまだという感じであり，相変わらずロットリングとインスタントレタリングが活躍している。清書用に使う場合の問題点をいくつか述べてみると，まず目盛りが枠の外側方向に出ており内側に向けられない，目盛りを打つ間隔が等間隔のみでありユーザーによる指定が行えない，などいくらでもある。データの値の絶対値が千を越えると，必ず「単位・千」というキャプションがついてしまうのもはなはだ迷惑な話だ。その他キャプションに上付き・下付き文字を使用できない，データ点の表示にエラーバーをつけられないなどの弱点がある。





一定値I_0に段々近づいていく関数$I(t)=I_0 \cdot (1-e^{-(t-t_0)/T})$に従うデータをそのままプロットしたグラフ(上)と，tとLOG(I_0－I(t))とでプロットし，最小二乗法で求めたI_0，t，Tの値を使って直線を当てはめたグラフ

しかし，なによりも清書用に使えない原因というのは，縦軸のキャプションが縦書きになることだ。日本語を使うときはまったく問題ないのであるが，英数文字の縦書き，こればっかりはいただけない。もともとのIBM-PC版ではきちんと90度回転した文字を表示しているのに。AXマシンの登場に伴い，IBM-PC/AT用ソフトの日本語対応化が続々となされて行くことを考えると不安でならない。

理工系ユーザー向けのソフト

ひと昔前は，表集計やデータベースのソフトなどはプログラミングを知らない人が使うものだという偏見があった。実際，市販ソフトもその程度のものが多かった。しかし実用的なワープロソフトが数多く登場したことにより，そんな時代は過ぎ去ったと言えよう。エディタアセンブラから始まったソフトウェアの統合化は，ワープロと図形処理，表集計とグラフ作成を結びつけてきた。メモリ限界はOSの進化により突破されようとしている。C言語を内蔵したワープロの登場も間近いのではないか。

◉特集

実践C言語からの誘惑

多くのプログラマが最も魅力を感じるプログラミング言語，それがC言語です。C言語は，UNIX記述用言語としてPDP-11というミニコン上で開発されたものであり，その生い立ちからUNIXと切り離して考えることはできません。しかし，C言語はUNIXの世界に留まらず，汎用コンピュータのシステム記述用として，そしてパーソナルコンピュータのアプリケーション開発においても高い生産性を発揮しています。最近ではパソコン雑誌でC言語に関する特集が組まれることも珍しくはなく，入門書も数多く出揃ってきています。
そんななかでOh!Xでは初めてC言語の特集をお贈りすることになりました。もちろん，Oh!Xでやるからには真っ正面からプログラミング言語としてのCを捉えてみたいと考えています。ここで注意してほしいのは，私たちは決して「これからはCの時代だ」と，大上段に構えたり，「皆さんもCでプログラムを始めましょう」とそそのかすつもりではないということです。C言語はいかにも話題のプログラミング言語ですが，誰にでも手軽に入門できる言語というわけではありません。それでも，C言語について学ぶことはコンピュータへの理解を深めるうえで意義深いことでしょう。単に使用法や書式を知るのではなく，C言語の思想を正しく理解することが大切です。言葉が文化を創るように，コンピュータ言語はシステム環境を創るからです。

Cのおかれた状況は？

パソコンのプログラミング言語として望ましいものは？　と考えたとき，果たしてC言語はどのような位置にあるだろう。C言語がパソコン雑誌などで騒がれるようになって5年以上になるが，確かに16ビット以上の機種では，Cが主力開発言語として確固たる地位を築いているようではある。しかし，これはソフトハウスや一部のマニアックなユーザーのレベルの話であって，一般の98ユーザーなどがCを使っているということではもちろんない（そもそも彼らの多くはプログラミングなどしない）。

BASICは，その名のとおり初心者を主な対象とした言語だから，プロの開発者がBASICからCに移行したからといって，それは開発者のレベルの向上でしかない。少なくともパソコンの標準言語として，今後CがBASICにとって代わるということではないはずだ。

ところが，アセンブラとの関係で見た場合，この先Cがアセンブラに代わる役割を担っていくことはかなり予想できるだろう。X1やMZなどの8ビットマシンではどうしたってアセンブラのほうが最終的に有利であったが，十分なメモリ，正しい最適化といった条件が揃えば，アセンブラにできてCにできないことはほとんどないように思われるからだ。

皆さんCはお使いですか？

さて，そのような状況のなかで，ユーザーはCに対してどのような見方をしているのだろうか。ちょっとOh!Xの若手スタッフに声をかけてみた。

——Cを使ったことがありますか？

▶ないです，でも早く使いたいなーなどと思っています。理由ですか？　そりゃ高級言語がアセンブラの代わりになるなら楽だからですよ。(で)

▶機会があれば使ってみたい。流行の言語だし，知っていて損はないと思う。(H.K)

▶使ってますよ。だってUNIX上じゃみんなCですよ。(A.K)

▶少なくとも私の周りにはCのできる人はいません。私が知っているのは，美しい構造化もできるし，汚いスパゲティもできるということ。(お)

やはり Oh! Xのスタッフでも，関心はあるが未だに使ったことがないという人が多いのが現状のようだ。

——CはBASICに次ぐ主流言語になり得ると思いますか？

▶思いません。パソコン買っていきなりCなんて無理ですよ。(で)

▶16ビット以上ならともかく，8ビットではメモリやスピードの関係で無理があるのでは？ (S.K)

▶流行は続くでしょうが，当然ROMにのっかることはないでしょう。それにCって難しいんでしょ？ (お)

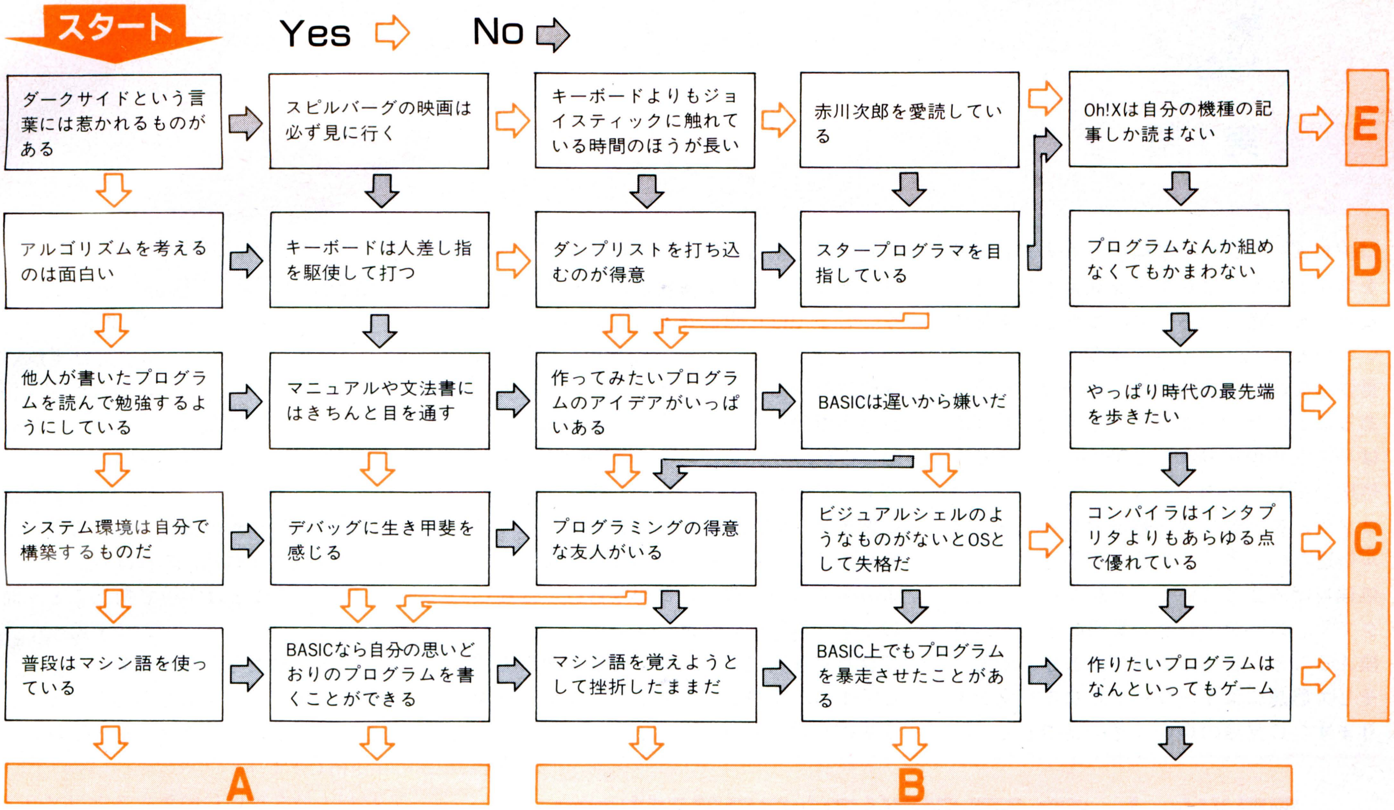
▶なり得ないと思う。一般民間人がお役所仕事を嫌うのを見てもわかるように，人間はあとに利益を得ることができるとわかっていても，その前の手続きを面倒に思うものなのだ。(く)

たいていの人が，これは無理という見方をしているようだ。まずはひと安心というところだろうか。

C言語入門適性チェック

というわけで，なにしろOh! X 始まって以来のC言語特集である。読者諸君ですでにCを使っている人はさっさと各記事を読んでいただくとして，まだCを使ったことのない人は手始めに，ここでC言語入門適性チェックを受けることをお勧めしたい。

フローチャートの出題とコメントは，村田敏幸氏にお願いした。なかなか意地悪な検査になっているが，まあ気楽に読んで楽しんでいただきたい（当の村田氏にもかなり思想的な偏りがあるようだが……，おお怖わっ）。



タイプA

このタイプのユーザーは強い。自分のパソコンは自分流に使うという，パソコンユーザーとしての基本的な資質を持っているといえるでしょう。はっきりいってCだろうが，アセンブラだろうが心配することはありません。というわけで，ここらで一発Cしてみますか？

8ビットマシンのユーザーでしたらCP/M上の安価なもの，ランゲージシリーズのC（αC）かSmall-Cあたりから始めるのが無難でしょう。これらはあくまでサブセットであり実用にはなりませんが，雰囲気をつかむには十分です。Cで新しいプログラミングのスタイルを修得したらアセンブラとの両刀遣いでマシンの限界に挑戦するというのもよいでしょう。

X68000ユーザーであれば，迷わずXCを買いましょう（どうしてもCP/M68Kが欲しいというのなら止めはしませんが）。

そしてやるからには，どちらの人もバイブルである『プログラミング言語C』（共立出版）を用意してください。不安であればもう1冊入門書を買ってもかまいません。あとはその入門書とバイブルとマニュアルを読みながら，小さくても「完結した」プログラムをたくさん作っていけば，段々とCの作法にも慣れていくでしょう。地道な努力があなたにはいちばん効果的な方法でしょう。

タイプB

素質はありそうですから，思い切ってCに挑戦するのもよいでしょう。が，まともに取り組むにはちょっと力不足のようですね。

もしも途中で挫折してしまったら，と不安を感じるようなら背伸びしてはいけません。あきらめて身を引いてください。BASICさんとお幸せにね，というところかな。

いやいやBASICもいいもんですよ。エディタから何から揃っているし，実行中マズイ！　と思ったらいつでもBreakできるし，何よりいつでも手軽に使えるし，その気になればかなり複雑なことだってできるし。こんな素敵なプログラミング言語が使えるなんて，あなたは幸せな人です。いーな，いーな，うらやましーな。

……と，この文章を読んで，バカにされているような気になった人にはまだチャンスはあります。今回の特集を徹底的に読みまくるなどして実力を磨き，Cの世界へ奇襲攻撃をかけましょう。

タイプC

フフフ，なんにも考えていないでしょ。あぶないなあ，このタイプの人はなにかと勘違いに見舞われやすいんですよ。感化されやすいというか，本に書いてあることを鵜飲みにして，自分でもよくわかっていないのに「～はちょっとねぇ」とか話しちゃうタイプ。金田正一がパ・リーグの試合を解説しているかのような，的外れなことばかりいう。なんだって？　自分はそうじゃない？　うーん，重傷だ。

もっとも，このタイプの人の中にはとんでもない天才がいるというケースもあったりするもんだから，ちょっと困ってしまうのだが。

まあ，ただの勘違い男だろうが，本当の天才だろうが，ノリの軽さこそがこのタイプの本質だから，ごじゃごじゃ考えずにCでもDでもやってみるのがいいかもしれない。勘違いというのはそれはそれで幸せなことだし，もしも天才であったら，本当に一発当てられるかもしれないから。ね。

タイプD

プログラムがちゃんと動いて，かつ出来がよければCで書かれていようがBASICで書かれていようがおかまいなし。喜んで使わせていただきます。

というのがこのタイプ。プログラミングとは無縁の生活を送っているのでしょう。かといって本当は自分でもプログラミングできればいーな，と思っているのではありませんか？　ただ，あと1歩が踏み出せないだけなのですよね。いろいろな本を読むだけ読んでわかったつもりになり，いつでもできると思って放ったらかしにしてしまうのかもしれませんね。それとも「こりゃ自分には無理だ」と思って投げ出してしまうのでしょうか。どちらにしろ，一度とことんまでプログラミングに浸かってみるべき時期が近づいているようです。試しにBASICで100行程度のプログラムを書いてみることを勧めます。それを完成させることができればよし，できなくてもそれはそれでよしとしましょう。しかし，完成できなければ，あなたにはプログラムを作るのに必要な何かが欠けているものと思ってください。

タイプE

忠告します。ここはさらしものにするために用意されたタイプですから，考え直してはどうでしょう。とりあえずは，Oh!Xをすみずみまで読んでからもう一度挑戦すれば，B，C，Dのどれかのタイプに行けることと思います。

それでも，「やっぱり僕はこのタイプだなあ」と思う人，少なくともこれからは，読むもの，見るもの，聴くものに注意しましょう。赤川次郎はいけません。あれは感性を擦り減らします。せめて，村上春樹を読みましょう。マンガなら「ぼのぼの」，アニメならば「Mr.味っ子」あたりがお勧めです。くれぐれもオタクと誤解されないよう注意しましょう。もちろん感性を磨くなら，やはりOh!Xがいちばんだと思いますが。

えっ？　Cへの適性はどうなのかって？　んなもんどうこういえる立場ですか。どうだっていいでしょう。わっ，怒ったあ？　冗談だってば。

◉特集 実践C言語からの誘惑

第1部 入門C言語の巻

コンピュータの初心者は，ともすると言語を学ぶということを文法を覚えるということと同義に考えがちです。しかし，実際にいろいろな言語を触ってみると，もっとも重要なのはその言語の思想といった部分であることに気づきます。たとえば「LISPはリスト処理が得意でCOBOLはファイル処理が得意」なのではなく，これらではプログラム自体がリスト処理であり，ファイル処理なのです。その言語の思想に従って，処理を組み立てていくことがプログラミングです。コーディング（実際の命令に置き換えること）自体は本筋ではありません。そして思想にはキーワードとなる概念があります。C言語の場合，その特徴として関数型言語である，データ構造が単純かつ柔軟であるといったことが挙げられるでしょう。ここではC言語をまったく知らないという人を対象に，こういった基本概念を集中的に解説します。しかし，これだけでC言語の思想に迫ることはできません。本格的にCを始めようという方は『K&R』を一読することを強くおすすめします。

C言語はBASICのように甘口の言語ではありません。だからといって，C言語は言語の中でも特に難しいというわけではないのです。具体的にイメージをつかみ，そしてひたすら実践。これはどんな言語でも同じですね。実践編に進む前にここでしっかり基礎を固めておきましょう。

関数とC言語“破門”講座

関数型言語として，システム記述用言語として，そしてアセンブラに代わる強力な能力を持つコンパイラとして注目の言語C。果たしてその実態はなんでしょう。あのゲーマー清水和人がFORTRANプログラマとしての意地をかけてCの本質に迫ります。

Shimizu Kazuto 清水　和人

まったくCがはやっている。中身は泥臭いくせに外見がなんとなくアカデミックで，しかも素人っぽくない。そういう外面がよいところが妙に受けているという言語である。

かのダイクストラの神様がのたまったような「構造化プログラミング」に向いているので一般ユーザーとは無縁に近い学者からも絶賛されるようになってしまったのだ。その上，アセンブラレベルの細やかな作業も高級言語の複雑な制御構造も併せ持っているので，まさに地上最強の暗殺言語としてパソコン界にもジワジワと根を伸ばしているのである。

しかし，Cは本来関数型言語であり，こんなメジャーになるような明るい言語ではなかったはずである。なにしろ一般のC言語の入門書などには必ずといってよいほど次の一文が入る。

Cはベル研究所でUNIX用に開発された言語である

というものだ。

ベル研究所というのは，アメリカでも超がつく一流研究所であるが，こんな説明を見て喜んでいちゃあ困る。私なぞはこれを読んだだけで「うわあ，堅苦しい」と思わず2，3歩ピョンピョンと退きたくなるほどだ。しかしパソコンマニアのなかには，逆にそういう堅苦しさにピョンピョン飛びついていく人が多いからなあ。

で，ソフトのくせにこんなに固いC言語はさらに，

システム記述用言語である

などという暗さも持っている。

OSのようなものを記述するのに便利だよというわけだが，じゃあなぜ「Cは関数型言語である」なのか？　関数ってものはf(x)とかcos θとかそんなやつだよねえ。システム記述は関数でするの？　などとつらつら見ていくと，とんでもない！　私の目に映ったのは，とても関数とは似ても似つかないCの本質なのだった。いったいぜんたいどおこが関数型なんだよおおっ？　関数ってのはそもそも……。

関数とは？

関数といえば，洋の東西を問わずf(x)だ。これはもうフーテンの寅さんが渥美清だというくらいの常識。xは変数で，その値の範囲を定義域と呼び，プログラム用語では引数（ひきすうと読む）などという。f(x)はxになんらかの操作を施した値を返す。f(x)＝2xとすれば，これは2倍する関数で……，ってことは学校で習うこと。簡単にいえば，

何か値を入れたら，何か返す

という不気味な箱（Black box）なのだ。

さて，そろそろプログラムを出そう。

```
main( ) { }
```

これが，最も短いCのテキスト（ソースプログラム）だ。この例はCの本には本当によく出てくるミーハーなやつで嫌いなのだが，ついつい私も使ってしまったというていたらくである。

先ほどのf(x)＝2xとの対応でいくと，fがmain，( )は( )だが，xはこの場合はない。また，{ }のなかに＝の右側の2xなどを書くところだが，これまたない。それでも「どうだ関数型だろう」といいたいらしいのである。ほらね，イヤラシイでしょう。実はこれ，何もしないプログラムなのである。何もしないという割には，引数を入れる( )や定義を書く{ }が残っているのだからなんか間抜けな感じがするかもしれない。ちなみにこのプログラムは，X68000でRAMディスクなどを使用しないと，実行形式のファイルにするのに（つまりコンパイルするのに）およそ1分ほどかかる。

次の例を出そう。

```
main( )
{
      printf("Welcome to C.¥n");
}
```

うーん，やだなあこんなプログラム。これもCの入門書には必ず出てくるタイプなんだ。例によって引数のない( )つきのmainのあとに定義が書かれている。「なになに，mainとはprintf(……)をすることか」。

printfはC言語本体のコマンドではなくオマケでついてくる関数だ。関数から関数

を呼ぶと。でもってprintfの引数はなんと文字列“Welcome to C.”である。まあ，ここまでは和田アキ子に免じて笑って許してあげよう。しかし，仏の顔も３度まで。ここではprintfという関数は，

なんの値も返していない！

のである。ゲゲッ，なんだあ，ぜーんぜん関数とちゃうやん，といっても，

いや，何も返さなくても関数なのぢゃ

とＣはのたまいなさるのぢゃ。まさに，柔の頑固おじいちゃんそのもの，さすがはベル研，とこういうわけである。

イヤラシイところはまだある。printfの( )のなかをよく見ていると，ほらほら腹が立ってくるでしょう？　このprintfは画面に文字列を表示するわけだが，引数の最後の「¥n」ってのがあたしゃいやだね，というおばあちゃんも多いと思う。¥nは改行の意味なのだが，Ｃをこれからやる人は，何かのおまじないと思って必ず¥nをつけるようにしてほしい。Ｃの本質を見極める鋭いあなたならいずれ理由がわかるはずだ。

まだまだある。printfの行の最後の；は何者か？　これは文の区切りだってことだから，これもそんなものかと思って納得してほしい。それにしても，この{ }の位置のいやらしさ。Ｃでプログラムを書く人は，こういう風に{ }をつけると読みやすいというのだが，なんかヒラメの目みたいでカッコ悪いねえ。

さあ，これだけクドクドいえばそろそろ，

Ｃはなんてイヤな言語だ

という暗示にかかってくれたろう。それじゃあ，もうちょっと現実的なプログラムに入ろうか。

アカデミックやなあ

さっそくだが，リスト１である。まずは関数らしいところを紹介しなければということで，ぐっとレベルを上げて足し算にいってみる。

見てわかるとおり，iは2，jは3でf(i,j)の値をijに入れて，i＋j＝ijを表示するのがmain関数である。f(ii, jj)は，iiとjjを足して値を返す関数である。

情けないほど簡単だ，という人にはリスト２をお見舞いしよう。これはなんと，

$\log(1+x)\ (|x|\leqq 1)$

を求めるプログラムである。へっへ～，

再帰を使っている

んだよお。再帰ってえのは再び帰るの文字どおり，関数が自分で自分を使っているという機能でＣ言語の特徴のひとつ，うーん，アカデミック！

計算モデルは，

$$\log(x+1)=x-\frac{x^2}{2}+\frac{x^3}{3}-\frac{x^4}{4}+\cdots\cdots+\frac{x^n}{n}$$

で級数展開されているわけだ。もっともこんなアルゴリズムは遅いからパソコンのルーチンには向いていないけどね。

mainのxが引数で，ここでは0.5にしている。求める値は，log(1.5)になるわけだ。このプログラムはf(x)とg(x, i)という関数を定義している。g(x, i)のなかではfor(……)という記述があるが，これがBASICでいうfor～nextにあたるやつだ。ちなみにj＋＋というのはj＝j＋1という意味で，ここでは詳しく触れないが，Ｃならではのおいしい演算子である。また，val＊＝xというのはval＝val＊xの意味。ほかにもいろいろあるから，Appendixのリファレンスをあとで見ておいてほしい。

g(x, i)はxのi乗を計算している。f(x)はそれを使ってx^i/iを計算して，次のiを用いた計算のために，またf(x)を呼び出して……とまあ，なんやようわからんけど，f(x)のなかにまたf(x)が書けるところが偉大だといいたいわけですよ。

また，この例でもわかるように関数の定義のなかにforループのような制御文が書けたりするのも特徴だ。ほかにもif～else，while，switch～caseなどの構造化制御文がいろいろと使えるというわけで，Ｃの関数は，関数というより，

サブルーチンだ

といってしまったほうが気が楽になるのであった。

それではサブルーチンということで

なあんだ，Ｃはサブルーチン型言語かあ。そうなりゃ，わかりやすいってもんですよね。要するにサブルーチンを作ってどんどん使う，これがＣによるプログラミングというわけだ。実際X68000のXC（C compiler PRO-68Kとやら）にはX68000の機能をフルにサポートしたサブルーチン（ライブラリ）がドカンとついてくるんだよ。

ではここからはちょっと，XCを例に取って遊んでみよう。X68000以外のユーザーの方には申し訳ないのだが，これからのＣ言語のあり方を考えるうえで参考にはなるはずだ。

まずはリスト３である。いきなりグラフィックというわけで，内容を見ると，なあんだX-BASICのサブルーチンを呼んでいるのかと誰でも気づいてしまうわけだ。そう，

XCではX-BASICと同等のシステムのサブルーチンが使える

のであった。

つまり，X68000にとってのＣは，BASICサブルーチンを組み合わせて構造化プログラミングを行うためのツールだったりもするのである。とはいえ，これは実はすごいことである。X68000のソフト体系は，BASIC→C→アセンブラのラインでがっちり固まったようなものだ。たとえば，グラフィックを使って新しいサブルーチンを作ってしまったら，次はそのサブルーチンを使ってさらにすごいサブルーチンを作っていくこともできる。

お次はリスト４。乱数ルーチンrandを用いた酔歩のプログラムである。酔歩とは，ヨッパライのおじさんが，フラフラと歩いているところからつけた名だ。X68000の画面の色を変えながらpsetが行くのである。j＝1から30000までにしたが，無限ループにすれば簡単な環境プログラムにもなる。受験勉強に疲れた頭を癒すには，こんなプログラムをいくつも作っておくとよいのではないか。

Ｃなんて堅苦しく考えるとろくなことがないから，とりあえずBASICルーチンで遊ぶというのがお勧めだ。そうしているうちにＣのプログラミングを覚え，また発想も湧いてこようというもの。入門書の１ペ

リスト１　簡単な足し算

```
main()
{
    int i,j,ij;
    i=2;
    j=3;
    ij=f(i,j);
    printf("%d+%d=%d¥n",i,j,ij);
}
f(ii,jj)
    int ii,jj;
{
    return (ii+jj);
}
```

リスト２　再帰を利用した対数の計算

```
main()
{
    float x=0.5,f();
    int i=30;
    printf("log(%f)=%f¥n",1+x,f(x,i));
}
float f(x,i)
    float x;
    int i;
{
    float val,g();
    i--;
    if (i==0) return(0);
    val=(2.0*(i%2)-1.0)*g(x,i)/i;
    return(f(x,i)+val);
}
float g(x,i)
    float x;
    int i;
{
    float val=1.0;
    int j;
    for(j=0;j<i;j++) val*=x;
    return(1*val);
}
```

ージ目から着実に進んで行くというのは考えないほうがよい。

んじゃ音楽でもやってみよっか

それでは，X68000のFM音源を使ったプログラムを考えていこうじゃないか。ルーチンはX-BASICのルーチンをそのまま使っちゃえ。というわけでリスト5である。これが基本形だ。

まず，m_initでFM音源を初期化し，m_allocでMMLのトラックバッファを確保する。それをm_assignでチャンネルに設定し，m_trkでトラックにMMLを書き込む。最後にm_playで演奏というわけだ。MMLについてはBASICマニュアルや中森氏の連載を参照すればよい。

リスト6は和音を鳴らすプログラムだ。文字列で音の名を入れると，それを同時に鳴らす。もちろん音は8音まで。

ところで，さっきから気になっている人もいるだろう。リストの頭にある，

```
#include <stdio.h>
    ⋮
```

という部分である。これは，XCについてくるライブラリを使うよということだ。ためしに，XCのシステムディスクのINCLUDEというディレクトリでどんなライブラリが用意されているか見ておくとよい。

まあFM音源関係のルーチンは全部music.hというインクルードファイルで呼び出せるから，

```
#include <music.h>
```

でOKだ。ほかにも，basic.hやbasic0.hに便利なルーチンがいろいろ入っている。ただし，BASICのライブラリを使う場合にはコンパイル時にオプション指定でcc /Wとしないとうまくコンパイルできないので気をつけよう（自分がひっかかっただけの話だったりして）。

話がそれたが，リスト6に戻ろう。ここでは和音を表す文字データをポインタを使って渡している。ポインタというのは，そのデータが入っているメモリの番地のことで，*dataというように，*のついた変数のようなものがその番地を表している。実は“ ”で囲んだ文字列データの場合，この番地でしか渡せないので，受けるサブルーチン側でも，ポインタで受けてやらなくてはいけないのだ。このポインタというのがあるおかげで，Cではアセンブラに匹敵する柔軟なことができるのだが，逆に初心者には理解しにくいということになっているのだ。

さて，waon（和音）という関数の引数になっているdataというのが，この場合ポインタで，文字列“CEG”の先頭にある“C”のアスキーコードが格納されている番地を指すことになる。さあ，このへんに興味を持ったらあなたもネクラなCの戦士だ。胸を張ってベル研への道を歩んでください。

また，ポインタの意味なんて興味ないという比較的正常な方は，このプログラムのmain関数をいろいろと変えて遊んでみるとよいだろう。

次のリスト7はちょっと初心に帰って，自動メロディ発生だ。ルーチン名もBach（バッハ）とした。ここでも乱数が活躍している。もちろん自動作曲となると乱数ばかりではダメで，ある程度規則性を持ったなかで音を選ぶことが必要になる。音楽好きの人はぜひチャレンジしてもらいたい。

リスト3　点，円，ボックス，ペイント

```
#include <stdio.h>
#include <basic0.h>
#include <graph.h>
#include <music.h>
main()
{
     cls();
     screen(2,0,1,1);
     vpage(1);
     pset(110,110,3);
     circle(200,400,100,4,0,360,200);
     box(250,250,300,300,8,32768);
     paint(200,400,6);
     fill(150,150,200,200,5);
}
```

リスト4　PSETによる描画「酔歩」

```
#include <stdio.h>
#include <basic.h>
#include <basic0.h>
#include <graph.h>
#include <music.h>
main()
{
     int i,j,im,ix=384,iy=256,rand();
     cls();
     screen(2,0,1,1);
     vpage(1);
     for(j=1;j<30000;j++)
     {
          pset(ix,iy,i+1);
          im=i;
          i=rand()%30;
          if (i==0) ix++;
          if (i==1) ix--;
          if (i==2) iy++;
          if (i==3) iy--;
          if (i>3)
          {
               if (im==0) ix++;
               if (im==1) ix--;
               if (im==2) iy++;
               if (im==3) iy--;
               i=im;
          }
          if (ix<0) ix++;
          if (ix>768) ix--;
          if (iy<0) iy++;
          if (iy>512) iy--;
     }
}
```

リスト5　MMLを使う

```
#include <stdio.h>
#include <basic0.h>
#include <graph.h>
#include <music.h>
main()
{
     m_init();
     m_alloc(1,2000);
     m_assign(1,1);
     m_trk(1,"Q7 @34 V9    O4 T66");
     m_trk(1,"L16 CCCDEG8GFFAG8");
     m_trk(1,"L16 >G<DD4DE-E-DC2");
     m_play();
}
```

リスト6　和音を鳴らす

```
#include <stdio.h>
#include <basic0.h>
#include <graph.h>
#include <music.h>
main()
{
     int i;
     m_init();
     for(i=1;i<9;i++)
     {
          m_alloc(i,2000);
          m_assign(i,i);
     }
     waon("CEG     ");
     m_play(1,2,3,4,5,6,7,8);
}
waon(data)
     char *data;
{
     int i;
     for (i=0;i<8;i++)
     {
          switch(*(data+i)){
               case 'C':m_trk(i+1,"C");break;
               case 'D':m_trk(i+1,"D");break;
               case 'E':m_trk(i+1,"E");break;
               case 'F':m_trk(i+1,"F");break;
               case 'G':m_trk(i+1,"G");break;
               case 'A':m_trk(i+1,"A");break;
               case 'B':m_trk(i+1,"B");break;
               case ' ':m_trk(i+1,"R");break;
          }
     }
}
```

リスト7　自動メロディ発生プログラム

```
#include <stdio.h>
#include <basic0.h>
#include <graph.h>
#include <music.h>
main()
{
     int i;
     char *bach(),*otolen();
     char mdata[2000];
     m_init();
     for(i=1;i<9;i++)
     {
          m_alloc(i,2000);
          m_assign(i,i);
     }
     for(i=1;i<2000;i+=4)
     {
          strcat(mdata,  bach());
          strcat(mdata,otolen());
     }
     printf("%s¥n",mdata);
     m_trk(1,mdata);
     m_play(1);
}
char *bach()
{
     int i;
     i=rand()%13;
     switch(i)
     {
          case 0 :return " C" ;break;
          case 1 :return "C#";break;
          case 2 :return " D" ;break;
          case 3 :return "D#";break;
          case 4 :return " E" ;break;
          case 5 :return " F" ;break;
          case 6 :return "F#";break;
          case 7 :return " G" ;break;
          case 8 :return "G#";break;
          case 9 :return " A" ;break;
          case 10:return "A#";break;
          case 11:return " B" ;break;
          case 12:return " R";break;
     }
}
char *otolen()
{
     int i;
     i=rand()%3;
     switch(i)
     {
          case 0:return "  ";break;
          case 1:return "8 ";break;
          case 2:return "2 ";break;
     }
}
```

パソコンのCとは

このように，X-BASICのFM音源ルーチン（くどいようだが関数のことだよ）を使うような例で見ると案外初心者でもCは役に立つ。他のパソコンでいまひとつ一般ユーザーに普及しないのは，やはり「関数ライブラリが貧弱だから」なのだろう。グラフィック，サウンドなどと，いちいちI/Oを考えながらサブルーチンを組むような使い方では，難しくてせっかくの素晴しい威力を持つCも使いこなせない。その点，X68000の持つ機能を生かすライブラリの充実がXCの魅力となっているのである。

ところで，関数型といっても実はサブルーチンに毛の生えたような形をとるC言語がなぜシステム記述言語といわれるのか。もうおわかりかもしれないが，ハードウェアを扱うルーチンを自在に組み合わせてコンパクトなマシン語に落としてくれるからだ（比較の問題だよ，誰だいCで書くと大きくなるっていってるのは）。高速だし，キー入力も少ないし，使いこなせればいい言語なんだろうなあ。

プログラムの例を続けよう。今度はX68000ご自慢のスプライトを使ってみたい。スプライトのライブラリもX-BASICのルーチンとそっくりなのがある。これについては中森氏の「X68000BASIC入門」の第4回(1987年11月号)に詳しく載っているので参考にしてほしい。

ここでは簡単なパターンを定義して，それをさきほどの酔歩のように動かしてみた（リスト8）。これは環境ソフトに発展できそうだ。こうやっているとなにかX68000を制覇したような気分になってくる。

かまわず続けて，今度はリスト9。が，これはこのままでは動かない。一応おしゃべりをするようにしたいのだが，ADPCM用の音声データを入力しなくちゃいけないのだ。膨大な量のサンプリングデータを載せるわけにもいかないので，ここでは形だけの紹介としよう。

関数型言語のよいところは，細かいことを気にせずに，まず全体の機能を考えてから次第に細かくプログラムしていくというトップダウン方式がとれることだ。私は昔からトップダウンが好きで，何かパソコンにやらせたいと思うとメインルーチンだけ作ってみたりする。

たとえば，リスト10に挙げた3つなんかどうだろう。ひとつは冷蔵庫のなかを掃除するプログラム（バカだね，しかし），ひとつは自動作曲プログラム，最後のひとつはカップラーメン作成プログラムである。まあ，退屈でヒマなときがあったらこんなのを書いてみるのも一興だろう。不思議なことに書いたものを眺めていると，なんだか本当にできそうな気がしてくるものだ。

X-BASICはインタプリタC(?)

X-BASICのコマンドばかり使ってプログラムを組んでいると，まるでX-BASICがインタプリタ型のCなんじゃないかと錯覚を起こす。まあ，最初からそのつもりで作られたBASICだから当たり前なのだが，X-BASICをやっておけばCにも入りやすいということも見逃せない。だいたいBASICと同じライブラリは「BASICマニュアル」を見ながら作成するわけだから，まったくCだかBASICだかわからなくなる。

さて，今度はBASIC以外のライブラリを使ってみようか。んー，ちょっと固い話だが，ライブラリmath.hから数値演算に関する主な関数を取り出して紹介しよう。リスト11がそれである。

よく見ると，おおっ，やっと「関数型」の恩恵にあずかれたではないですか。printfのなかでそのまま関数の名を書いてやれば，FORTRANやBASICのようにわざわざ変数を使って値を返さなくてすむ（わかるかな？）。

ご覧のように逆三角関数（arccosなど）や双曲線関数（cosh）も入っている。X68000で数値演算をしたい人は，BASICなんか使っていたらどうしようもないよ。これはもうCでプログラムを組むっきゃない。

リスト8 スプライトを動かす

```
#include <stdio.h>
#include <basic0.h>
#include <graph.h>
#include <music.h>
     char data[255]=
     {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
      0,3,3,3,3,3,0,0,0,3,0,0,0,0,0,0,
      0,3,0,0,0,3,0,0,0,3,0,0,0,0,0,0,
      0,3,0,0,0,3,0,0,0,3,0,0,0,0,0,0,
      0,3,0,0,0,3,0,0,0,3,3,3,3,3,0,0,
      0,3,0,0,0,3,0,0,0,3,0,0,0,3,0,0,
      0,3,3,3,3,3,0,0,0,3,0,0,0,3,0,0,
      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
      0,0,0,0,6,6,0,0,0,0,0,6,6,0,0,0,
      0,0,0,0,0,6,6,0,0,0,6,6,0,0,0,0,
      0,0,0,0,0,0,6,6,0,6,6,0,0,0,0,0,
      0,0,0,0,0,0,0,6,6,6,0,0,0,0,0,0,
      0,0,0,0,0,0,0,6,6,6,0,0,0,0,0,0,
      0,0,0,0,0,0,6,6,0,6,6,0,0,0,0,0,
      0,0,0,0,0,6,6,0,0,0,6,6,0,0,0,0,
      0,0,0,0,6,6,0,0,0,0,0,6,6,0,0};
main()
{
     int i,j,im,ix=128,iy=128;
     cls();
     screen(0,3,1,1);
     sp_clr(0,255);
     sp_off(0,127);
     sp_def(0,data);
     sp_disp(1);
     for (j=1;j<30000;j++)
     {
          sp_move(0,ix,iy,0);
          im=i;
          i=rand()%15;
          if (i==0) ix++;
          if (i==1) ix--;
          if (i==2) iy++;
          if (i==3) iy--;
          if (i>3)
          {
               if (im==0) ix++;
               if (im==1) ix--;
               if (im==2) iy++;
               if (im==3) iy--;
               i=im;
          }
          if (ix<0) ix++;
          if (ix>256) ix--;
          if (iy<0) iy++;
          if (iy>256) iy--;
     }
     screen(2,0,1,1);
}
```

リスト9 未完のお喋りプログラム

```
     static char data[50][3900];
main()
{
     char dat[100];
     strcpy(dat[0],"     .....");
     pronou(dat);
}
pronou(dat)
     char dat[100];
{
     if (dat[1]=='ア') a_play(data[1],2,3);
     if (dat[1]=='イ') a_play(data[2],2,3);
     :
     :
}
```

リスト10 幻のメインルーチン3部作

```
/*   冷蔵庫の掃除   */

main()
{
     雑巾を洗う();
     電源を切る();
     ドアを開ける();
     for(i=1;i<6;i++)
     {
          i段目のものを出す(i);
          i段目を雑巾でふく(i);
          i段目のものをしまう(i);
     }
     ドアを閉める();
     電源を入れる();
     雑巾を洗う();

}

/*   自動作曲プログラム   */

main()
{
     拍子を決める();
     テンポを決める();
     調性を決める();
     長さを決める();
     for(i=1;i<長さ;i++)
     {
          和音を決める();
          メロディーを決める();
          他の声部の音を決める();
     }
     楽譜に出力する();
     演奏する();
     売り込む();
     著作権使用料を貰う();
}



/*カップラーメンの作成*/

main()
{
     お金を持つ();
     コンビニエンスストアへ行く();
     インスタントラーメンを選ぶ();
     レジへ行く();
     お金を払う();
     お釣りをもらう();
     家に帰る();
     湯を沸かす();
     ラーメンのフタを開ける();
     熱湯を注ぐ();
     食べる();
     片付ける();
     消化する();
     main();     /*   うわあ、再帰だあ   */
}
```

それでは，やっと関数らしい関数が出てきたところで，ちょっぴり変わったプログラムを紹介しよう。リスト12がそれである。おおっ，今度はmainの( )のなかに引数が入っている。そうです，このようにした場合，Human68kのコマンドラインから，

A>SAMPLE　5

のように引数を（この場合は5）を入力できるのであった。とっさに思いつく使い方としては「究極のプログラム電卓」。実験データをまとめてレポートにするときのややこしい計算もCがあれば大丈夫。ちょちょいのちょいというわけでプログラム電卓を買うお金があったらゲームソフトにでも回しましょう。

この例を見ると，引数が2つなのにコマンドではひとつの引数しか入れていない。これは第1引数には，引数の数が入るためだ。プログラムのほうの事情として，もし引数の数が違っていたら何かのエラー処理が必要なので，こういう規則になっているというわけだ。一見間違いやすいから注意が必要だろう。でも使いなれると文字列の形でも入るのでファイル名などの記述に非常に便利。まさにOSから直接呼べる関数なのだ。

うーん，そうか。やっとわかってきたような気がするぞ。OSから引数が渡せるからシステム記述に向いているんだな。Cを使っていろいろなコマンドを作っていくことができるじゃあないか。つまり，まず"コマンド名.c"というCのソースを書いてcc.x（コンパイル&リンク）にかければ，コマンド一丁出来上がりってことだ。

Cに明日はあるのか

以上，Cのさわりの部分をちらちらと眺めてきたわけだが，じゃあCにとって「明日はどっち」なのだろう。とはいっても，Cには構造体，共用体，記憶クラスなど，まだまだ難しい機能がいっぱいあるので，これだけのことから何かいえる筋でもないのだが，あえて考えてみよう。

ではリスト13を見てもらいたい。これは，test.datというファイルにキーボードから入力したデータを書き込んでいくプログラムである。男の勝負は一見してどう感じたかで決まる。

1)　うわあ，こんなの絶対わからん。見ただけでめまいがする。

→　こういう人は，はっきりいってCには近寄らないほうがいい。

2)　うっそー，わかんない。でも面白そう。

→　こんな人は，祝氏が始める連載講座に進んでください。

3)　なんだ，こんなの簡単じゃん。

→　あなたはどうしてこんな文章を読んでいるのか。さっさと，第2部に進んだほうがいいですよ。

これがすべての答えだと思う。リスト13の例のように，Cはファイルを扱うのに便利な機能を備えている。しかし，どうもとっつきが悪い。何か人を寄せつけない雰囲気がリストから漂ってくるではないか。その原因は果たしてなんだろう。

第1に，括弧の多さである。引数を囲む小括弧( )，手続きを囲む中括弧{ }，そして配列の添え字の大括弧[ ]と，これで高校があれば日本の教育制度である。だいいち，こんなリスト，声に出して読めないじゃない？　main( ){……を「メイン小カッコカッコ中カッコ……」などと読み出したら，思わず「静かな湖畔の森のかげっからっ……」と歌ってしまうではないか。

とっつきにくい2番目の理由は，省略形が多いこと。i++とかのインクリメント・デクリメント演算子を考えたのはセンスがあるが，どうもリスト全体の表現が簡潔過ぎてプログラムをわかりにくくしている気がする。

そして最後は私の嫌いなポインタという概念だ。プログラムが高度になれば，ポインタのポインタの配列だとか，もっと複雑な使い方もバンバン出てくるってんだからお手上げである。リストに*がひとつ抜けていただけで泣いたプログラマも多いのではないか。

ただ今回いろいろと試してみた結果わかったことは，そんな難しい機能は使わずに，とりあえずライブラリを使って遊ぶ気持ちでいれば，一変してなかなかにイイ言語になってくれるということだ。特にXCのライブラリの充実度は群を抜いている。X68000ユーザーには「この果報者！」といって鼻を指でつっ突きたいほどだ。だからCの本など読むのはやめて，まずいじくってみること。そこから道が開けようと悟った私であった。

めでたし，めでたし

いやあ，素晴しい。まさに完璧な言語でした。なんといっても関数型の特徴を実によく生かしていて，まさにシステム記述やOSのユーティリティ作戦にはもってこいですね。しかもライブラリを使えばハードの機能を生かすのも非常に簡単だし，ソースコードの打ち込みも省略形が多いから短時間ですみますよね。

だから最初に言ったじゃない。いい言語で僕も大好きだって。絶対メジャーになるって。え？　言ってない？　んなことないよ。錯覚錯覚。皆さんも絶対にCの勉強しましょうね。祝さんの連載も始まるし。CをやればBASICがイヤになるよ。ほんとに。やってて損はないよお。と，こう書いたら怒られるかしら？

リスト11　数値演算の例

```
#include <stdio.h>
#include <basic0.h>
#include <graph.h>
#include <music.h>
#include <math.h>
main()
{
     printf("arccos(0.5)=%f¥n",acos(0.5));
     printf("arcsin(0.5)=%f¥n",asin(0.5));
     printf("arctan(0.5)=%f¥n",atan(0.5));
     printf("cos(pai)=%f¥n",cos(pi));
     printf("sin(pai)=%f¥n",sin(pi));
     printf("tan(pai)=%f¥n",tan(pi));
     printf("cosh(0.5)=%f¥n",cosh(0.5));
     printf("sinh(0.5)=%f¥n",sinh(0.5));
     printf("tanh(0.5)=%f¥n",tanh(0.5));
     printf("exp(0.5)=%f¥n",exp(0.5));
     printf("log(0.5)=%f¥n",log(0.5));
     printf("log10(0.5)=%f¥n",log10(0.5));
     printf("2**10=%f¥n",pow(2.0,10.0));
     printf("root(2)=%f¥n",sqrt(2.0));
}
```

リスト12　コマンドの作成

```
#include <stdio.h>
main(i,j)
int i;
char *j[];
{
     if (i!=2) printf("error!!¥n");
     else
     printf("%s¥n",j[1]);
}
```

リスト13　謎のデータ入力関数

```
#include <stdio.h>
#include <fcntl.h>
#include <fctype.h>
#include <stat.h>
#include <io.h>
main()
{
     int fil,ilen;
     char dum[81];
     if ((fil=open("test.dat",O_TRUNC|O_WRONLY))<0)
          if ((fil=creat("test16.dat",S_IWRITE))<0)
               exit(0);
     while(1)
     {
          printf("¥ndata ?");
          scanf("%s",dum);
          strcat(dum,"¥n");
          ilen=strlen(dum);
          if(dum[0]=='/') break;
          if(write(fil,dum,ilen)<0)
          {
               printf("error!!¥n");
               break;
          }
     }
     close(fil);
}
```

データ構造からの"Hello C World"

初心者にとって最難関となるのがデータ構造でしょう。抽象的になりがちなテーマですが，具体的にどういった処理が行われているのかを中心にイメージを作りあげるように解説してみました。データ構造を理解することがC言語入門の最大の課題です。

Sohma Hidetomo 相馬 英智

Cとデータ構造の重要性

C言語はとってもおいしい言語です。おいしいとするにはそれなりの理由がいるわけですが，CはBASICに比べれば速いし（コンパイルするから当たり前），小回りがきくし，OSとは仲がいいし，複雑なデータ構造が扱えたりするところがおいしいのだろうと思います。その中でも特に広い意味でCのおいしさの60～70％ぐらいが，データ構造あたりにあるのではないでしょうか。そこで，ここではデータ構造という面からCについて考えてみましょう。

しかし，データ構造にはBASICにない概念も多いので，入門者にとって脱落者のもっとも多いところでもあるのです（特にポインタ関係）。入門者がデータ構造でつまずく理由についても考えてみたいのですが，これはパソコンでよく使われる入門者用の言語（特にBASIC）は一般にデータ構造が貧弱であるためと思われます。このため，BASICしか使ったことのないユーザーはデータ構造やこれを主体とした処理のアルゴリズムなどについては知識が不足気味のようです。しかし，データ構造を有効に活用するというプログラムテクニックはきわめて重要なものなのです。

PASCAL，CなどのALGOL系の言語ではこのことをとても重視して設計されています。これは，PASCALなどの言語の生みの親であるN.Wirth大先生がこのことをきわめて重要視したためです。現在パソコン関係の人なら誰でも知っているデータベースというものは，このデータ構造の考え方を大きく発展させ，複雑で膨大なデータを使いやすくし実用化をはかったものといえます。

入門者にとって，Cなどは変数の宣言すら面倒に見えると思います。しかし，変数の宣言ひとつとってもその意味にはかなり深いものがあります。またCでは比較的自由にデータ構造を使えるので，これを使いこなせば複雑な処理を比較的簡単に記述できるのです。

変数は語る

皆さんは変数をどんなイメージでとらえていますか？　もっとも一般的なとらえ方は，数値などを入れる入れ物のようなものだと思います。しかし実際コンピュータ内では，この入れ物に直接相当するものはありません。ではどうやって変数を実現するのかというと，コンピュータ内の記憶領域の一部（レジスタを含む）を変数と見立てているのです。

といってもマシン語を使ったことのない方には理解しにくいでしょうから，コンピュータの記憶構造から説明しましょう。

コンピュータには大きく分けて2つの記憶領域があります。ひとつは主記憶と呼ばれるもので，一般に半導体メモリ（メモリICなど）で構成されています。そしてもうひとつが補助記憶（装置）と呼ばれるもので，具体的にはディスクや磁気テープなどを指します。この両者の違いはデータの読み書きに必要な時間（アクセスタイムといいます）と，どのくらい多くのデータを格納できるかという記憶容量に表れます。その構成部品の特性によって，主記憶はアクセスタイムが短く高速にデータの読み書きが可能であるが，価格の面から記憶容量を大きくできません。これに対して，補助記憶は安価に大容量にできるのですが，アクセスが低速になってしまうのです。

したがって，コンピュータでは頻繁に使うデータを主記憶に置き，あまり頻繁に使わないデータを補助記憶などに置くことでデータのアクセスタイムを短く抑え，処理能力を向上させています（これを記憶の階層化といいます）。

パソコンなどの小型のコンピュータの場合，主記憶はメモリチップのみで構成されるのでメモリ空間などということもあります。しかしここでは，私の独断で主記憶空間といういい方で統一します（本当は，どっちがいいのだろう？　でも，難しい本は主記憶空間っていうのよね）。

さて今までの話から想像がつくと思いますが，一般には頻繁に使われるデータを格納する変数はプログラムの高速化のために主記憶上に作られるのでした。そこで注意しなければならないことは，現在のコンピュータの多くがストアードプログラミング方式（ノイマン型）というやり方をとっているということです。この方式は主記憶空間にデータであろうがプログラムであろうが一緒に置いてしまおうというものです。

厳密にいうとOSなどが，データとプログラムをある程度区別して扱ってくれたりします（あくまでもある程度だよ）。しかし，主記憶空間に置いてしまえば，どちらも単なる1と0が並んだだけのもの（1と0のビット列）となってしまいます。したがってOSが馬鹿な場合は特に，どれをデータとし，どれをプログラムとするかは実際に処理するユーザープログラム次第となってしまうのです。

これは一歩間違うとプログラムを破壊したりする可能性を秘めていますが，狭い主記憶空間しかない場合などは，プログラム自身でプログラムを書き換えることでプログラムを有効に使えるようにすることが重要な意味を持つことがあります（16ビット機以上では百害あって一利なし）。とにかく大事なことは主記憶空間にデータもプログラムも存在してしまうんだよということです。

では次に主記憶がどのような構造をしているかを見てみましょう。主記憶空間は8ビット（1と0だけの2進数が8桁，まとめて1バイトともいいます）ごとの小さな番号のついた箱の集まりとみなすことができます。この番号をアドレスといい，0から順番に主記憶の量だけの値をとります。たとえばあるコンピュータ内の主記憶容量が64Kバイトだと，一般にコンピュータの世界では，

K＝1024（$=2^{10}$）

なので，

64Kバイト＝65536＝10000_H

つまり，16進で10000_H個の箱があることになります。その箱にはそれぞれアドレスがあるのですが，アドレスは0から始まるので0000_Hから$FFFF_H$までの番号があることになります。この箱を人の住んでいる家と見なせば，このアドレスは番地のようなものです。そこで，このアドレスという言葉は日本語では番地と訳されたりします。

そこで主記憶空間は1次元となっていて，イメージ的には図1のように連続した帯の

図1　メモリのイメージ

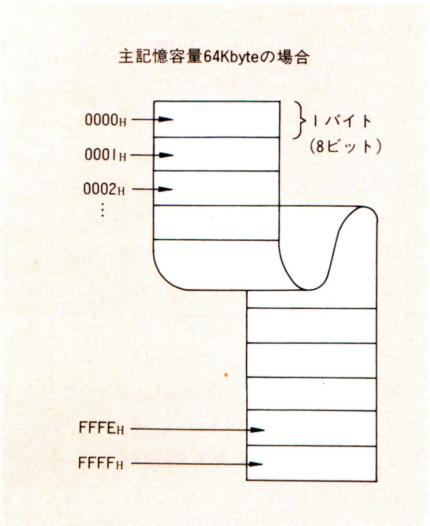


図2　変数のイメージ

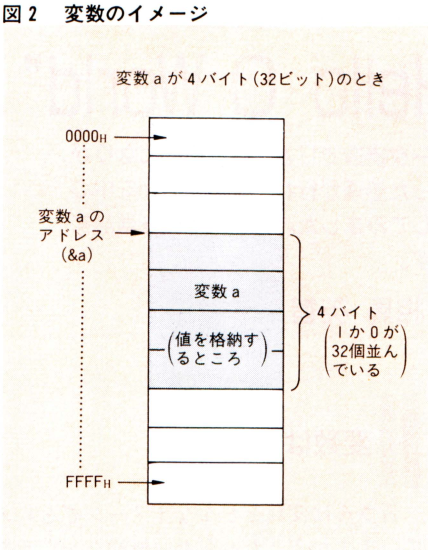


表1　基本データ型(XCの場合)

型	サイズ	表現できる値の範囲
char	8ビット	−128〜127
int	32ビット	−2,147,483,648〜2,147,483,647
short int	16ビット	−32,768〜32,767
long int	32ビット	−2,147,483,648〜2,147,483,647
unsigned char	8ビット	0〜255
unsigned int	32ビット	0〜4,298,967,295
unsigned short int	16ビット	0〜65,535
unsigned long int	32ビット	0〜4,298,967,295
float	32ビット	10^{-37}〜10^{38}
double	64ビット	約10^{-307}〜10^{308}

ようにみなすことができます。Cでは，この主記憶空間のイメージが絶対必要ですので，しっかり頭に叩き込んでおいてください。

さて変数の話に戻りましょう。主記憶空間が1バイトごとに区切られているというのは物理的な（機械がどうできているかという）話で，我々はこの1バイトにあまりとらわれずに記憶空間を使うことができます。

しかし，コンピュータ自体がこの1バイトを基準として処理を行うように設計されていますので，具体的には1〜数バイトをまとめて変数として使用します（Cの場合,多少の例外があったりします)。つまり,記憶空間の連続した一部分（数バイト）を大きな箱に見立てて，変数としています。したがって，変数は必ず自分が記憶空間のある位置としてのアドレスまたはそれに類する値を持っています。これをイメージ的な図にしたものが図2です。

多くのプログラミング言語では，この変数のアドレスを具体的に知ることや主記憶から直接データを読んだり書いたりすることなどは難しくなっています。というのは,主記憶空間にあるのは変数のみではなく，プログラム自身やコンピュータ内部の重要な情報がたくさん蓄積してあるためです。万一これらが書き換えられるとコンピュータが異常な動作をしかねません。しかしCではこれをポインタというものを使って簡単に知ることができます。これは危険なことですがこれによってマシン語で書かないとできないようなことまで可能となり，うまく使えばかなり面白いことができます。

変数と型

次に変数の型というものについて考えてみましょう。変数に格納する値にはさまざまなものがあり，数値，文字，文字例（文字が集まったもの）などを基本として，数学的な論理値，集合などが考えられます。しかし，コンピュータ内部ではすべての値が2進数の数字で（つまり1と0だけが並んだビット列で）表現されているのでした。たとえば，文字はキャラクタコード（またはシフトJISコード）と呼ばれる番号が各文字につけられていてこの番号で表現されています。

そこで各変数を使うときには，そこに格納するデータの内容についてコンピュータに知らせる必要があります。Cなどのプログラミング言語ではその領域に格納されたビット列を数値として解釈するのか，文字として解釈するのか，それともほかの値として解釈するのかを変数を宣言するときにコンピュータに知らせます。

また，格納するデータによって変数そのものが記憶空間に実現される領域も変化します。具体的にいうとCでは型によって変数が何バイトとなるかというサイズが異なっています（参考までにXCの型とサイズを表1に示します)。つまり，変数を宣言するということは変数を型に合わせて記憶領域に生成し，その変数に名前を（Cでは変数や関数などの名前を総称して識別子といっています）つけることを意味します。さらに，Cでは記憶クラスといって変数を具体的にどのように生成するかを指定することができたり，変数の生成を言語の処理系にまかせず，我々が直接変数を生成できるような（実際は記憶領域を確保する程度）関数も準備されています。これらを使うにはそれなりのテクニックが必要なので最後に述べることとして，まずは標準的なCのデータ型について見ていきたいと思います。

Cの基本データ型

Cのデータ型は大きく分けて基本データ型と，それらの組み合わせで作られる構造データ型（これには正式な名前はないようです）と，あの高名な（悪名高き？）ポインタがあります。そこでいきなりですが，基本データ型について見ていきましょう。

基本データ型はその名前のとおり，きわめて基本的なデータ型で複雑な構造などといったものを持ちません。一般的にCには整数型，浮動小数点型の2つがあり，最近の傾向としてはこれらに列挙型,void型(関数の帰り値の型のみに有効）を加えたものが標準になりつつあります。しかし，Cのサブセットなどでは列挙型やvoid型はもちろん浮動小数点型すらサポートされていない場合がありますので注意してください。

それでは整数型から見ていきましょう。整数型は文字どおり数値のうち，整数を格納する変数です。どうして，数値のうち整数だけを特別扱いするのかというと，プログラムの中でこの整数の処理が群を抜いて多いということが挙げられます。そこでこの整数だけを扱う変数を作ることで処理の高速化などをはかろうというわけです。

先ほどコンピュータ内のすべてのデータは1と0のビット列でしか表現されないとい

う話をしました。問題はそのビット列の取り扱い方なのです。実は単純に数値ひとつをとっても，その表現のしかたはいくつもあります。そこで整数型は一般に2の補数表示という表現のしかたを用いて値を格納しています。このやり方は整数の格納に向いていますが大きな数の格納には不向きです。しかし，整数の処理が単純で高速化がはかれるという利点があり，多くの言語の整数型がこの表現方法を採用しています。

一般に整数型には，int型とchar型があります。int型は整数を格納するためのもので変数の型としてはもっとも頻繁に使われるものです。通常，longとshortがあり，格納できる値（整数）の最大値が異なります（これについてXCの場合を表1に示します）。

int型だと格納する変数の範囲が狭いという人がいるかもしれませんが，Cの整数型は処理速度の向上を狙って設計されているという感じが強くなっており（一般に16ビット以上のマイクロプロセッサではアキュムレータと同じ大きさになっています），そのためあまり範囲は重視されていないようです。

どうしても大きな値を格納したいときは，のちに述べる浮動小数点型を使ってください。これは，格納する値の範囲や精度を重視したものとなっていて，その分速度が犠牲にされており，処理が遅くなることはいうまでもありません。

また，char型は文字型と呼ばれるもので整数型と区別されることもあります。これは文字（1文字）を格納するためのものですが，実際はキャラクタコードつまり整数値として処理していますので整数型といったほうが適切だと思います。ここでいう1文字というのは半角1文字のことで，要するにキャラクタコード表に載っていない文字（コード）は格納できません（全角文字を格納するには2バイト必要です）。

さて，Cでは文字は‘ ’でくくることで対応するコードの値として示されます。例としてchar型の変数cに文字Aを代入してみましょう。

```
char c;
c ='A';
```

無論，複数の文字を一度に‘ ’でくくることはできません。

さらに最近はint型，char型にunsignedが指定ができるようです。このunsign指定をすると変数の取り得る範囲が変わり，正の値しか使用できなくなりますが，正の値が倍の範囲だけ使えるようになります。つまり整数型は通常，値を2の補数表示という数値の表現方法を用いて値を格納していますが，unsign指定をすると絶対値表示という表現方法を用いるようになります。

Cではビット処理を行うことがあるのですが，このときに2の補数表示を使っているとまずいことが起こります。そこで代わりに絶対値表示を使ってしまおうというものです。この指定は，もともとビット処理を行うときにパワーを発揮するもので通常はあまり使用しません。

次に浮動小数点型ですが，これには整数や小数などの数一般の値を格納するもので（無論，変数内部では浮動小数点表現を用います），floatとdoubleの2つがあります。floatとdoubleの違いは格納された値の精度が異なることです。いうまでもなく，doubleを用いたほうが高い精度の値が得られます。しかしCの浮動小数点型は演算結果に得られる精度を重要視していて処理速度は遅くなっています。はっきりいって，オマケみたいなものです。長い目で見てやってください。

そして最後に列挙型，void型ですが，これらについてはサポートしてないCがあることなどから今回は割愛させていただきます。それでは，これらの基本データ型を用いて形成される構造データ型について話を進めたいと思います。

構造データ型

これまでの話はデータ構造の中でもきわめて基本的な話で，これ以降の話が本当にデータ構造らしい話となります。それにともない，話も少し難しくなるかもしれません。

今まで話した基本データ型は変数そのものについての話で，各変数にはひとつの値しか格納できませんでした。また格納された値同士の関係といったものは全然表現されませんでした。しかし，これから説明する構造データ型は基本データ型の変数を組み合わせることで変数に構造を与え，データの値だけではなく，そのデータ間の関係や構造までも表現してしまおうというものです。

実際に私たちのまわりに存在する情報を見回してみると，情報がひとり立ちしているわけではなく，情報同士はもちろんのこと，いろいろな要素と複雑にからみあっています。そこでその情報（データ）を格納する変数に構造を与えることでデータの持っている特性を無理なく吸収し，表現してしまおうというのがデータ構造の狙いなのです。

ここまで話せばデータ構造の重要性と，それがしっかりしているプログラミング言語がいかに優れているかということがわかってもらえるものと思います。

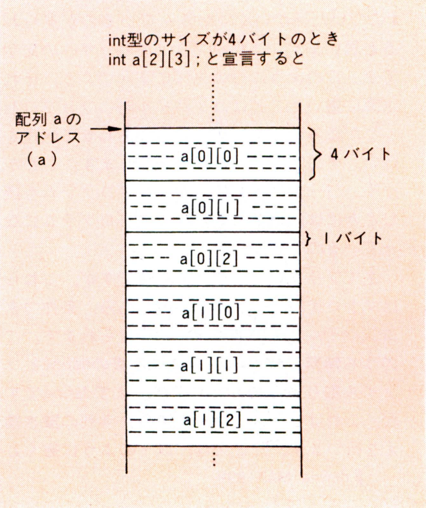
ただ注意すべき点はデータを処理するのはコンピュータなので，コンピュータにとって自然で無理のないデータの表現で処理効率の向上や高速化をすることが重要です（無論，得られた結果を人間が理解しやすいようにすることも重要です）。そこでしばしば，私たちにとって理解しがたいデータ構造を用いることがあります。しかし今のところ特殊な例を除くと，極端なまでに複雑なデータ構造はあまり効率がよくありません。むしろ，単純なデータ構造をうまいこと組み合わせたようなものがよい結果を生み出すことが多いようです。したがって複雑なデータ構造といっても，それは基本データ型に比べて複雑という程度ですので安心してください。

さてCの構造データ型には，配列，構造体，共用体が用意されています。それでは，それらについてひとつずつ見ていきたいと思います。

毎度お馴染みの配列

まずBASICでもお馴染みの配列です。配列とは，ある特定のデータ型の変数を複数集めて，ひとつの変数（データ構造）とするものです。同じ型で表現された値を複数格納できますので，区別をするために格納する位置に番号をふります。この番号を添え字といいます。実際に配列を宣言したときの主記憶上にとられた配列の状態を図

図3　配列の実際



3に示します。

これを見ればわかるように同じ型の複数の変数を連続した領域に取ったように見えます。そこで，配列のとる記憶容量は要素の型のサイズの要素数倍となっていて，単純に計算することができることがわかります。Cの構造データ型などの場合，実際に使う変数（データ構造）がコンピュータ内でどのように表現されているかを知っておくことは，きわめて重要であると思います。しかしマシン語が使える程度の知識までは要求しませんので，安心してください（これでもCは一応，高級言語ですから）。

さて，Cで配列を宣言するときに注意してほしいことは，使用する添え字が必ず0から始まるということです。したがって，以下のようにint型の配列aを宣言すると，

```
int a[5];
```

配列aの要素数は5個で，添え字は0から始まるので0から4までの5つが使えることになります。また，多次元の配列を使うときは，各次元ごとに添え字を'['と']'の大括弧でくくります。たとえば2次元の配列bを宣言して，値を代入するときは，

```
float b[5][5];
b[3][2]=1.5;
```

というように書きます。これらのところがほかの言語と異なりますので注意してください。

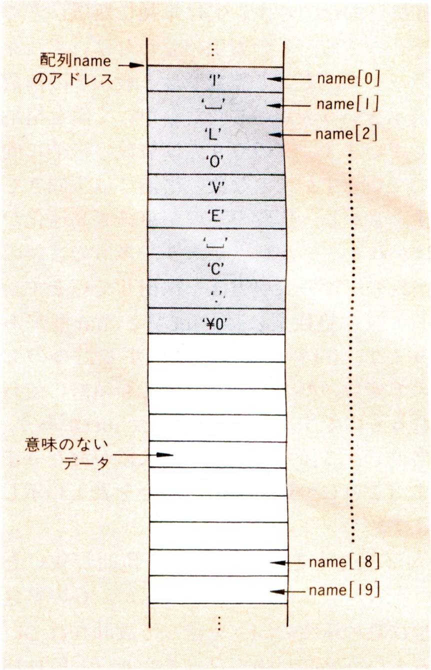
さて，Cにはchar型（文字型）と呼ばれるものはあるのですが，文字列型というものはありません。文字列とは，複数の文字が連なったデータのことをいいます。というと難しそうですが，我々が文字のデータを使うときは1文字だけ使うよりもいくつかの文字を組み合わせて使うことが多いはずです。そこでCでは，この文字列をchar型の1次元配列として実現しています。具体的には，宣言された配列の小さな添え字のものから1文字ずつ格納していって，最後の文字を格納したあとにヌル文字（Cでは'¥0'と表現されます。なおこのヌル文字は1文字です）と呼ばれる文字を格納します。このヌル文字はここで文字列のデータがおしまいだよということを示すものです。こうすることで文字列を格納するとき，あらかじめ配列を格納する文字列より大きめに取っておきさえすれば問題なく処理が行えます。

とはいえ，Cは文字列処理は得意ではありません。というのは文字列の処理を行おうとすると，演算子などでは処理が不便なので文字列処理関数をincludeして使用することになるからです。たとえば，nameという配列に"I LOVE C."という文字列を格納してみましょう。

```
char name[20];
strcpy(name,"I LOVE C.");
```

上に出てきたstrcpyという関数は文字列処理用の関数で，1番目の引数に文字型配列をとり，これに2番目の引数の文字列を格納するというものです。この処理の終了したときの配列nameの状態を図4に示します。これを見ると宣言時に20個の要素がとられ，格納した文字列の最後にヌル文字（'¥0'）があり，配列内のこれ以降のデータは意味のないことを示しています。

図4　文字列の格納



このように基本的に1バイトごと（char型変数1個ごと）に1文字（半角）を格納します。しかし日本語の使える処理系では，漢字などの全角文字も格納することになります。この場合，全角文字は2バイトで表現されるので文字型の配列要素2個に1文字格納されることになります。

さて先ほどの例では文字列の値を配列に格納するのに，strcpy関数を使いました。そこで文字列型が配列で実現されることからわかると思いますが，以下のように書くことはできません。

```
name ="I LOVE C.";
```

どうしてもこのように書きたいときは次のように書きます。

```
char *name;
name ="I LOVE C.";
```

これはあとで説明するポインタ型を用いたものです。これは変数に特定の文字列（文字列定数）しか与えないときにはよいのですが，通常の文字列処理用の関数が使えなくなる場合があります。

また配列を宣言するときに，要素数が最低でも格納する文字数＋1になるように注意してください。1文字多いのは，エンドコードとしてヌル文字を格納するためです。

プリプロセッサは偉大である

一般にC言語の処理系（この場合はコンパイラ）には，純粋にコンパイルする前にソースプログラムに手を加えてくれるプリプロセッサというものがあります。このプリプロセッサはプログラムに細かな細工を施してくれる便利なものです。

実際にCのプログラムを見てみると，先頭に'#'のついた行がいくつか並んでいるのが見られると思います。これがプリプロセッサへの命令です。このように行の先頭に'#'がつくと，それ以降に書いたことはコンパイル時にプリプロセッサが読み取って，その内容にあわせて処理を行い，その'#'のついた行を削除します。したがって，パーサ（構文解析部）がプログラムを読むときには，通常の完全なCのプログラムになっているのです。

さてプリプロセッサへの命令ですが，これらはマクロ定義，ファイルの取り込み，条件つきコンパイルとその他に大きく分けられます。このうち頻繁に使うのがマクロ定義のdefineとファイル取り込みのincludeです。まずdefineですが，これはソースファイル内の文字列の置き換えを行います。たとえば，以下のように書くと

```
#define MAX 999
```

そのソースファイル内のMAXという文字列がすべて999と書き換えられます。したがって，まるで定数を宣言したかのように書くことができることになります。ほかにも単純な関数であればあたかも，それを定義したかのように振る舞わせることもできます。

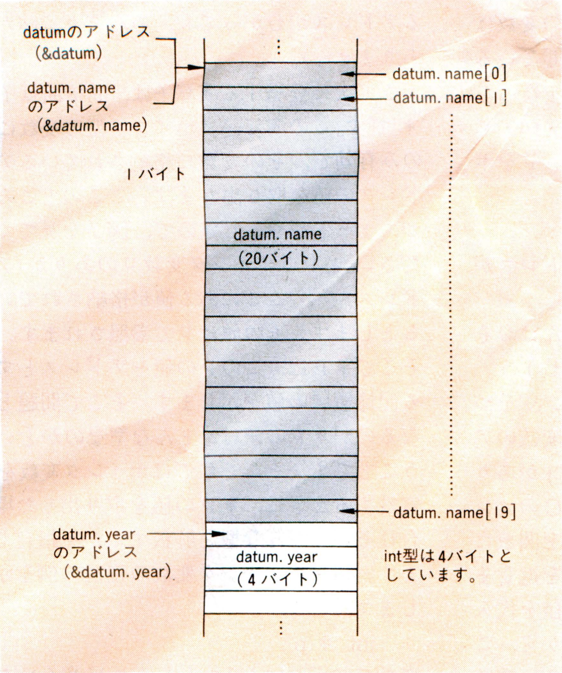
そして，もう一方のincludeは指定したファイルの取り込みを行います。

```
#include <stdio.h>
```

よくプログラムを見ると上の例のように，関数についての情報が書かれているインクルードファイルを取り込んでいますね。上の例はプログラム中で標準I/O関数群を使用する場合の宣言です。このようにCのプログラムでは，あらかじめ用意しておいたプログラムをライブラリとして使うことができます。そのおかげで我々は複雑なプログラムをいちいち書かずにすみますし，ハードの仕様変更にもライブラリの交換だけで対応できるのです。

以上のようにプリプロセッサを使うとプログラムがかなり書きやすくなります。また，プリプロセッサにはこれらの例のほかにもいろいろな命令があり，これを真面目に説明しようとすると本が1冊書けるぐらい奥が深いのでした。

図5　構造体の例



さて,先の例で文字列を“”でくくっていることに気づいたでしょうか。そう，Cでは文字列は“”でくくって示されます。“”でくくると，くくられた文字列の最後にヌル文字を加えたものを表します。したがって，‘a’と“a”では意味が異なります（‘a’は1文字，“a”はヌル文字が加わった2文字となります)。Cでは文字と文字列の区別は重要なので，注意してください。

まとめてしまおう構造体

次に構造体です。配列がある特定の型の変数を集めてひとつの変数にしたのに対し，構造体は複数の型の変数を集めてひとつの変数とするものです。その際，構造体の各要素のことをメンバといいます。そして配列では各要素を添え字で指定していましたが，構造体では構造体宣言時に各メンバにメンバ名という名前をつけてあげて区別します。以下に年齢格納用のメンバyearと名前格納用のメンバnameを持つ構造体，datumを宣言してみましょう。

```
struct person {
        char name[20];
        int year;
} datum;
```

上の例ではpersonといった文字が見えます。これはタグと呼ばれるもので，上で宣言したnameとyearのメンバを持つ構造体の総称名です。これに対して，datumはpersonという構造体の型を持った変数の名前です。このタグ名を設定することで，ある1種の構造体のメンバについての情報を何度も書く必要がなくなります。このタグ名と変数名は両方とも必ず指定しなければならないという規則はありません。そこで先の例は以下のようにも書けます。

```
struct person {
        char name[20];
        int year;
};
struct person datum;
```

ではこのように宣言された構造体はどのように主記憶上に実現されるのでしょうか。上の例で宣言された構造体datumの場合を図5に示します。配列の場合と同様に，構造体内のメンバが連続した領域に複数の変数を配置したようになっています。当然，構造体の記憶容量（サイズ）は各メンバのサイズの合計となっています。したがって，各メンバのサイズがわかれば簡単に構造体の大きさが得られます。各メンバにはその型に合わせてふつうの変数のように値が格納されます。

さて各メンバに値を代入したりするときは，構造体名の後ろにメンバ演算子“.”をつけてメンバ名を指定して使います。先ほど宣言したdatumの各メンバに値を代入してみましょう。

```
datum.year=70;
strcpy(datum.name,"Richard Feyn
man");
```

また，各メンバの値を参照するときも同様に行えます。実際さまざまのデータを処理しようとすると，この構造体のような複数のデータ型が集まったような複雑なデータが多くあります。しかし，構造体だけではあまりおいしくありません。ところがあとで説明するポインタ型とこれを組み合わせるときわめて強力です。私などはこれができるというだけでCやPASCALを使っているようなものです（PASCALではrecord型といっています)。

構造体にはメンバの特別な型としてビットフィールドというものがあります。これは各メンバが整数型でunsigned指定された場合に表現でき，その表現する値の範囲が小さい場合に有効となるものです。このようなときに各変数の記憶領域における大きさを必要最小限とすることにより，従来ひとつの変数の領域としてとる場所に複数の変数を詰め込んでしまおうというものです。これはCによってはサポートされていない可能性がありますので，詳しい説明は省略させていただきます。

なんでもしまえる共用体

そして，最後に共用体です。これはひとつの変数に，複数の型を持たせてしまおうというものです。つまり共用体は同一の記憶領域に複数の変数があるようなものです。したがって構造体では各メンバにそれぞれ値を格納することができたのですが，共用体では常にメンバのうちのひとつだけが有効となります。そこで重要なことはこの共用体を使うとき，どの型で共用体変数に値を格納したか（どのメンバが有効か）をユーザーが覚えておく必要があるということです。最初に述べましたように変数の型の宣言とは，その変数の領域に格納されたビット列をどのように解釈するかということの指定なのです。したがって，共用体変数に格納した型と異なる型で値を取り出そうとすれば，意味のない値を得ることになります。

共用体の宣言の方法は構造体とまったく同様となっています。以下に共用体の記述例を示します。

```
union TEL {
        int no;
        char mess[15];
} Richard;
```

すると構造体と同様に，

```
Richard.no=1102956382;
```

図6　共用体のイメージ

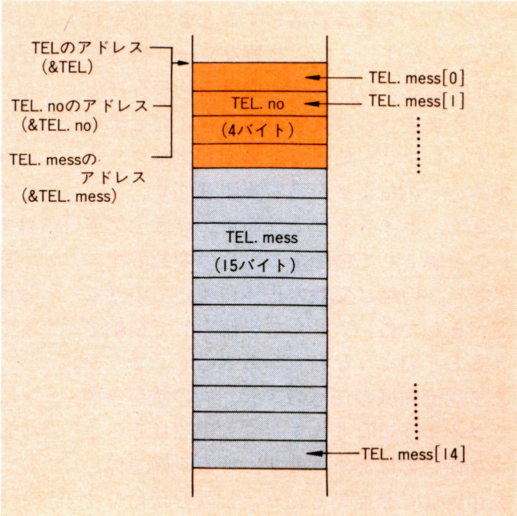
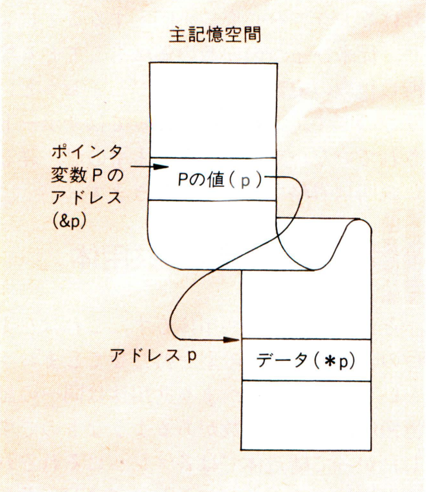


図7　ポインタによるアクセス



とか

　　strcpy(Richard.mess, "He has no TEL.");

などと書けるようになります（両方一度に書くと，あとに書いたものだけが有効となります)。つまり，メンバ名を指定することにより型を指定しています。

上の例として宣言した共用体TELがどのように主記憶空間に実現されるかを示したものが図6です。これを見ればわかるように共用体の場合，メンバの中でもっともサイズが大きい型だけのサイズが共用体自身の記憶容量（サイズ）となります。また各メンバのアドレスは共用体自身のアドレスとなっています。したがって，複数のメンバのうち常にひとつしか使えないことになります。

くどいようですが，値を取り出すとき，型に十分に注意して使ってください。

やっぱり，ポインタ型は無敵だ！

ポインタ型は，きわめて便利な変数のひとつでデータ構造の話をするときには不可欠なものです。また，複雑なデータ構造を記述するときの縁の下の力持ち的な存在ともいえます（私は接着剤的な存在だと思います)。しかし，使い方を誤るとプログラムが暴走しかねませんので要注意です。

さて，ポインタ型とは簡単にいうとアドレスを値として格納するための変数です。アドレスの話は変数の正体についての話で述べましたね。アドレスを値として格納できるとどのような利点があるのでしょうか。まず記憶領域の任意のアドレスの値が参照できるようになります。でも実際はそのような使い方はあまりせず，変数や構造体，共用体などを連結したり，配列を使いやすくしたりすることによく使われます。

先ほどいいましたように，通常の変数はその変数が記憶領域の一部を占領しているので，その位置すなわちアドレスを持っています。変数の実際にある位置の先頭アドレスを取り出す演算子が，'&'です。pをポインタ変数として宣言すると，

　　p=&a；

とすることで，pには変数aのアドレスが格納されます。

C言語を使ったことのある皆さんはおそらくscanfという関数をご存じでしょう。この関数を使うときに，変数の前に'&'をつけて関数を呼びますね。これには変数のアドレスを値として渡そうという意味があります。もし'&'をつけなかったら，Cのコンパイラはその変数に格納された値を関数に渡し，その値のアドレスに戻り値を返します。しかし実際には，その変数に値を代入して関数を終了してほしいのですから，ここに値を代入するんだよというために変数のアドレスを渡しているのです。たとえるなら「Oh！Xに返信用封筒つきで手紙を書いたら返信用封筒がSTUDIO Xに掲載されてしまった」ようなものです。

Cでは関数はひとつの値しか返しません(もともと関数とはそういうものですが)。しかし，複数の値を返したいときがあります。この場合は関数を呼ぶときに変数のアドレスを渡してやり，関数内でこれをポインタ型で受けてやればよいわけです。こうすると呼び出し側の変数と，呼び出された関数内の変数がひとつに結びつけられたように見えます。

このような変数の渡し方をアドレス渡し(call by address)といい，一般的な，変数の値をコピーして関数内の変数に渡してあげるやり方を値渡し(call by value）といいます。ここで重要なのはアドレス渡しの場合，関数内で変数の値を変更すると呼び出し側の変数の値も変わってしまいます(したがって，関数から複数の値が返せる)。しかし，値渡しの場合は呼び出し側の変数と関数内での変数は関係ないので，関数内で変数の値を変更しても呼び出し側に影響はありません。

話がそれてしまいましたが，ポインタには直接任意のアドレスの値を代入することもできます。たとえば，

　　p=100；

といったぐあいです。この場合この100という値はアドレスを示すものと解釈されます。逆にポインタ変数のアドレスを取り出すこともできます。つまり，&pとすればよいわけです（でもポインタ変数の実現されているアドレスがわかっても，あんまり意味がありません)。

さて次に登場する演算子は，間接演算子'*'です。この演算子はポインタ型の変数にのみ有効で，その変数に格納されているアドレスの値を取り出します。ここのところがちょっとややこしいので，図7を見てください。&pはポインタ変数pのありか(アドレス)です。ここにある値が格納されているとします。その値はpで参照されます。そこで*pは，このpの値をアドレスとする記憶領域の値を示します。そこで問題となるのが*pの値は，どんな型なのかということです。そこでCではポインタ変数を宣言するときに，*pの値を取り出した場合にそれをどんな型と解釈するかを指定します。以下にポインタ変数の宣言の例を示します。

　　int *p；

変数名の前に'*'がつくとそれがポインタ変数であることを示し，上の例は，*pの値をint型とすることを意味します。そこで，次のように書いたとします。

　　int a, *p；
　　p=&a；
　　*p=5；

するとpは変数aのアドレスを示すため，aの値が5になりこれは次の1文と等価です。

　　a=5；

これは，えらくまわりくどい例で，このようなことはあまりしません。しかしポインタ変数のイメージはわかってもらえたと思います。

そもそもポインタとは日本語でいう矢印のことです。つまりポインタ変数はアドレスを値として格納することにより，主記憶領域の一部分を指し示しているわけです。

このポインタ型を使うときに注意してほしいことがあります。それはポインタ型の変数を使うときは必ず初期化をして使うということです。たとえば，以下の例を見てください。

　　int *p；
　　*p=0；

このようなプログラムは非常に危険です。というのは，ポインタ変数pの値（アドレス）がわからないということです。したがって記憶領域のたまたまpで示されたアドレスに値を書き込んでいるわけで，そこがなにか別のプログラムに使われていたり，ましてや自分自身のプログラムだったりすると暴走や異常動作の原因になります。

それから，Cにおいてはすべての文字列

はポインタ型であるということです。文字の配列のところで一度説明しましたが，たとえばプログラム内に“I LOVE C.”という文字列があったとします。すると，コンパイルされた結果得られるオブジェクトプログラム内には，この文字列がそのまま格納されています。したがって，この文字列のアドレスを使えば文字列処理用の関数の手を借りなくてもすみ，以下のように書けます。

```
char ＊mess；
mess＝“Message.”；
printf(“％s¥n”,mess)；
```

これは，決まった文字列（文字列定数）のときしか使えませんが，同じ文字列を何回も書かずにすみます。

ポインタ型と構造データ型の不思議な関係

今までの説明でポインタ型がなんとなくわかっていただけたでしょうか，しかしポインタ型のおいしさは，配列や構造体などと組み合わせて使うときに生きてきます。そこで，ポインタ型と配列の関係について見てみましょう。

配列の各要素を参照するときは，添え字を用いるのが一般的ですが，ポインタ型を用いてもできます。まず，以下のように配列とポインタ変数を準備したとします。

```
int a[20],  ＊p, i；
p＝&a；
```

すると配列aの要素を参照するにはa[i]と書く方法と，＊(p+i)と書く方法があり，これらの書き方はまったく同じことを意味します。つまりポインタ変数に1を加えるということはその要素の型分だけアドレスを増やすことになり，したがって配列の次の要素を示すようになります。これは1を加えるときだけでなく，ほかの値を加えるときも差し引くときも同様に行えます。ふつうはこの配列aの20個の要素を値の合計を得るためには，以下のように書きます。

```
int total
total＝0；
for (i＝0;i＜20;i＋＋)
        total＝total＋a[i]；
```

しかしこれはポインタ変数を用いて，以下のようにも書けます。

```
int total,n；
total＝0；
n＝20；
while (－－n＞0)total＋＝＊(p＋＋)；
```

実は，配列の[ ]の括弧も演算子で，計算機内部でも同じようなことをして処理をしています。

さて大事なことは，Cでは変数(基本データ型）は常に2つの値を持っているということです。つまりひとつは変数に格納された値で，もうひとつは変数の実現されているアドレスです。ふつう変数から値を取り出そうと変数の名前を書くと，変数に格納されている値が取り出されます。これに対して変数にアドレス演算子‘&’をつけることでアドレスを値として取り出すことができます。このことについては今までの話でわかってもらえたと思います。

それでは構造データ型のものはどうなっているのでしょうか。少なくとも配列はポインタ型であると考えられます。つまり配列の名前はその配列の格納されている先頭アドレスを示しています。そのアドレスにそれに対して[ ]の括弧を使った演算子がついて，先頭アドレスからのずれを配列要素のサイズと[ ]でくくられた数から算出し，求めるアドレスを得て，それに対して処理を行っているのです。つまり，以下の2つの宣言文は同じことを意味します。

```
int ＊p；
int p[ ]；
```

このような書き方はCでは比較的よく行われます（特に文字列処理の場合）ので，できるだけ慣れておくのが好ましいでしょう。逆にいえば，このような書き方が簡単に読めるようになるとCについては十分に使えるレベルに達したといえるのではないかと思います。ただし，このような書き方をするとき注意しなければならないことがあります。それは演算子の優先順位です。これが自分の組んだプログラムとかみあっていないと，予想外の結果が得られることとなります。おかしいなと思ったらこの辺を疑ってください。なお，優先順位を（強引に)変える方法は( )でくくってやることぐらいしかありません。

それでは，構造体とポインタ変数の関係へと話を進めましょう。構造体の場合は配列の場合と異なり，配列のように表現を変えるといったものではなく，構造体同士または構造体と変数の連結を行いデータ構造を形成するためのものです。具体的には，ポインタメンバ演算子‘－＞’を使用します。以下に，構造体のところで宣言した構造体タグpersonを用いた例を挙げましょう。

```
struct person datum；
```

最近のCの傾向と対策

いきなりだが，Cは言語だったりするのです。したがって生きています。そうトンボだって，ミミズだって，乳酸菌だって生きているのです。ですから生きている限りは進化するのが人情というものだったりするのでした。そこで，C言語も徐々にではあるが進化しているんだよというわけです。しかしC言語だって初めからこのような姿であったわけではないのです。有名な話ですが，もともとBCPLという言語があって，これが進化してBとなり現在のCが生まれたというルーツがあります。

CはあのK&R(カーニハンとリッチーの書いた『プログラミング言語C』という本）で定義されていて，これにはずれるものがCと名乗ることは難しいのです。ただし，Cの機能を少し削って実現したサブセットというのはあります。BDS Cを始め，8ビット用のCの多くはサブセットです(αCはBDS Cのさらにサブセット)。

そしてK&Rが出たあと，Cはデータ型を増やしたり(void型やenum型が有名)，関数の増強をはかったりしながら変化していきました。そして現在，Cは変革の時期にあります。もともとCの移植性がよいという伝説は，CがUNIXの標準言語であり（UNIXにはひとつのCしかない)，マシンが違ってもUNIX上ではわずかな変更だけで同じものが動作することからきていました。ところがUNIX以外の処理系でもCが広く使われるようになり，いわゆる「UNIXのC」以外のCが多くなったのです。もともとK&RでCのすべてが定義されているわけではありません。細かな部分や曖昧な部分は，実際の処理系移植者の判断に委ねられているのです。そうしてK&Rの解釈の違いから，いろいろなCが現れ分化し始めました。

そこで「標準的なCを決めよう」ということでプログラミング言語の統一化をはかっているANSI（アメリカ規格協会）が乗り出しました。これが，ANSIの標準化案（ANSI仕様)であり，関数使用の際にパラメータの型をチェックすることにしたり，構造体同士の代入や関数が構造体を値として返すことなどを可能とすることを盛り込んでいます。お気づきの方も多いと思いますが，XC（というよりもMS C）はこのANSIの標準化案を取り入れています。現在，多くのCがこの仕様に準拠するための拡張をはかっているところです。

そして，もうひとつの傾向がオブジェクト指向との融合です。これが，C＋＋という言語です。C自体が優れた言語だっただけに，このC＋＋は次世代言語としてかなり期待されています。言語とはいっても，実際にはこのC＋＋は一種のプリプロセッサのようなもので，C＋＋で書かれたプログラムはCのプログラムに変換され，それをCでコンパイルし実行します。したがって，Cの動く環境ではたいがいC＋＋を使うことができるでしょう。このあたりはCと同様に実用重視の設計となっています(もっとも，数行のプログラムをコンパイルしたら数十Kバイトのオブジェクトを出したとかいう話もありますが)。

```
struct person * data;
data=&datum;
data->year=70;
strcpy(data->name,"Richard");
```
と書くことで，従来のように処理ができます。つまり，
```
datum.year=70;
strcpy(datum.name,"Richard");
```
と同じことをしていることになります。実は，ポインタメンバ演算子を用いずに書くこともでき，それは以下のようになります。
```
(* data).year=70;
strcpy((*data).name,"Richard");
```
要するに，この書き方はポインタメンバ演算子と等価といえます。

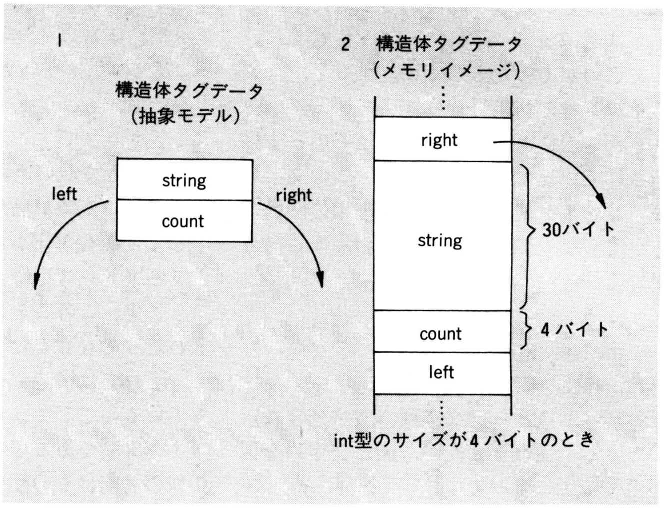
このポインタメンバ演算子とポインタを用いると，構造体を組み合わせて新しいデータ構造ができることを意味します。このことは非常に重要です。

しかしこのポインタメンバ演算子を用いても構造体は配列のように，ポインタでも同様に処理ができるかというとそうではありません。先ほど配列はポインタであることを述べました。しかしその特性からいうと，構造体はどちらかといえば配列よりも変数に近いようです。厳密にいうと，変数に近づきつつあるというのが正しいでしょう。Cが生まれた頃は，構造体の使用は処理速度の低下を招くため，配列のような取り扱いを行っていました（現在でも多くのパソコンのCはやっています)。しかしこれからのCはANSI標準化案に対応していく過程で拡張され，これにともない構造体は変数のような特性を帯びていくものと思われます。事実XCはANSI標準化案に準拠しており，このためほかのCより構造体がかなり使いやすくなっています。

秘奥義 記憶クラス

今までの話は変数やデータ型の特性についてのものであるのに対して，記憶クラスはCの処理系にどのように変数を生成するのかということを指定するものです。この変数の生成方法を指定するということは，変数の特性の大部分を指定することともいえます。実際にはCには自動変数，静的変数，外部変数，レジスタ変数があります。では，これらについてひとつずつ見ていきたいと思います。しかし，ここのところはかなり難しいので，理解できない人も多いと思います。そのような人は以下を読み飛ばしてもかまいません。しかし，わからないからといってCが使えないわけではありませんから，あせらずに勉強してください。

図8　構造体タグデータ



自動変数はCではもっとも一般的な変数であって変数宣言時（関数宣言内）になにも指定しないと，この自動変数になります。以下に宣言の例を挙げます。
```
auto int a;
float b;
```
この変数は宣言されたときにスタックに生成され，宣言文のすぐ前の'{'に対応する'}'のところまでが有効範囲で，'}'のところでその変数は消滅します。また，生成されたときにすでに同名の変数が存在したときは，その変数より優先します。以下に例を示します。
```
main( )
{
    auto int a;
    {
        float b;
        {
            int a;
            a=1;
        }
    }
}
```
いちばん最後に宣言された整数変数aは，その前後の括弧{}内で有効で，1はこの変数に代入されます。このように自動変数は，自分の寿命が最小になるように生成され，消滅します。これは，モジュール化などをやりやすくし，プログラムの安全性を高めるためです。

次に静的変数ですが，この変数はプログラムが実行される際に生成されプログラムが終了するときに消滅します。しかし，プログラムの最初に宣言したとき（外部的静的変数）と，関数内で宣言したとき（内部的静的変数）で働きが異なります。この外部的静的変数は，宣言を行ったソースプログラムファイル内のすべての領域から参照代入ができます。これに対し内部的静的変数は，その宣言をした関数内でしか参照代入ができません。以下に変数の宣言の例を示します。
```
static int a;
```
さて，この静的変数は生成時に初期化が一度だけ行われ，初期化の値を指定しないときは0で初期化されます。またプログラムの最初のところで記憶クラスを指定せずに変数を宣言すると静的変数になります(自動変数としても変数の有効範囲がプログラム全域で，1回しか初期化されないので静的変数とほぼ同一となります)。

またCでは関数は別々にコンパイルが可能であり，ソースファイルが複数になることがあります。このようなときに変数をソースファイルの最初のところ（厳密には関数外ならよい）で，静的変数宣言をするとほかのソースファイルからその変数の参照はできません。このことは変数だとあまり意味がありませんが，関数を静的に宣言することでほかのソースファイル内からの呼び出しを禁じることができ，プログラムの安全性を向上させることができます。

静的変数は同一ソースファイル内の複数の関数間で同一変数の参照を可能にしたものでした。しかし外部変数は複数のソースファイル間の同一変数の参照を可能とします。具体的にはいちばん最初にこの変数が宣言されたときに変数の生成を行い（これは静的変数となります)，それ以降に宣言されたとき（extern宣言を行う）は同一の変数となるようにします。以下に例を示します。
```
int com;
main( )
{
……
}
extern int com;
sub( )
{
……
}
```
この中の変数comは2つの関数でバラ

バラに作成されていますが（同時にコンパイルされていなければならない），実際には同一変数となります。

最後にレジスタ変数ですが，これは変数をレジスタにとることで，処理の高速化を行おうというものです。しかし，CPU の持ちあわせるレジスタの個数は有限なので，やたらとレジスタ変数にしたからプログラムが速くなるかというとそうではありません。優れたCの処理系（コンパイラ）になると，レジスタ変数で宣言しなくてもプログラムの高速化をはかるために頻繁に使う変数は自動的にレジスタ変数にするものもあります。このような処理系の場合，レジスタ変数の宣言をしてもレジスタに変数を生成してくれないものもあります。以下にレジスタ変数の宣言例を示します。

register int a；

このように記憶クラスを使うと変数の特性を変えたり，複数の人数でのプログラムの開発をやりやすくしたりすることができます。しかし，変数などがどのように実現されているかということをよく知っていないと，これを使いこなすことはできません。このことは結構レベルの高いテクニックですので，最初のうちはあまり必要でないでしょう。しかし，大きなプログラムを作成しようとするときなどには大いに役に立つものと思います。

縁起もののサンプルプログラム

話はここで終わってもよかったのですが，この手の話にはサンプルプログラムが縁起ものなので，構造体とポインタを組み合わせたデータ構造を用いたプログラムをリスト１に示します。このプログラムはかの有名なCのバイブルともいうべきK&R（『プログラミング言語C』）という本の中のプログラムを手本に書いたものです（なんで書き直したかというと，ここのテーマがデータ構造なのでそれ以外のプログラミング要素を省くためです）。それでこのプログラムは入力された文字列を内部に蓄えておいて，文字列ごとの出現回数を表示します。はっきりいってあんまり意味のないプログラムです。

さてこのプログラムで使われているデータ構造は，二進木(binary tree) と呼ばれる有名なものです。リスト１の構造体タグdataを宣言しているところを見てください。これが二進木の構成要素です。これのイメージを図8-1に示します(この図は今までのメモリイメージのものとは異なります)。また，メモリイメージで書くと図8-2のようになります。いうまでもなく矢印はポインタを示しており，二進木の特徴は要素のポインタが右と左の２方向を同時に指しているということにあります。これからはいちいちメモリイメージで書くと面倒なので，図8-1のような抽象的なモデルで考えます。実はここがデータ構造のしっかりした言語のおいしいところのひとつで，いつもメモリイメージで考えていたのでは，面倒臭くて頭がこんがらかってしまいます。しかしこのような抽象的なモデルでデータ構造を表現できればさほど難しくありません。このように抽象的にデータをとらえられたことは，あのオブジェクト指向などでかなり重視されています（その点からいうとCのデータ構造は，あまりおいしくないような気がします。Cのデータ構造がおいしいというのはあくまでもBASICなどの言語に比べてだったりするのでした）。

リスト１　二進木のサンプル

```
/*    二進木によるサンプルプログラム   */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define STRLEN 50
#define TRUE 1

struct data {
        struct data *right;
        char string[STRLEN];
        int count;
        struct data *left;
};

static struct data *root;

main()
{
        char input[STRLEN];
        struct data *insert();

/* 初期設定 */

        root = NULL;
        printf("¥nこれは、二進木によるサンプルプログラムです。¥n");
        printf("数えたいものの名前をいれてください。¥n");

/* メイン処理 */

        while (TRUE) {
                printf("名前を入力してください。  ");
                scanf("%s",input);
                root = insert(root,input);
                search(root);
        }
}

/* 木のデータを探索し、すでにあればカウントを増やし、無ければ付け加える。 */

struct data *
insert(p,input)
struct data *p;
char *input;
{
        int flag;
        if (p != NULL) {
                if ((flag = strcmpi(p->string,input)) == 0 ) ++p->count;
                else if (flag > 0) p->right = insert(p->right,input);
                else if (flag < 0) p->left  = insert(p->left ,input);
        }
        else {
                if ((p = (struct data *)malloc(sizeof(struct data))) == NULL) {
                        fprintf(stderr,"メモリが不足しました。¥n");
                        exit();
                }
                strcpy(p->string,input);
                p->count = 1;
                p->right = p->left  = NULL;
        }
        return(p);
}

/* 木のデータを小さい順に表示する。 */

search(p)
struct data *p;
{
        if (p != NULL) {
                search(p->left);
                printf("%s : %d¥n",p->string,p->count);
                search(p->right);
        }
}
```

図9　二進木

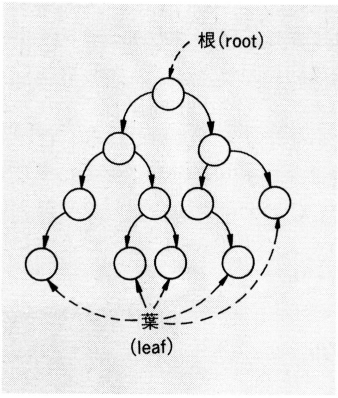


図10　二進木のデータ関係

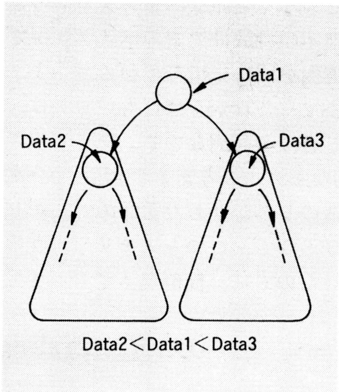
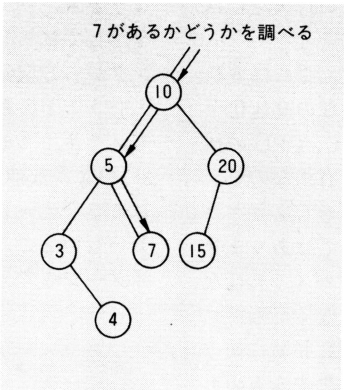


図11　二進木による検索



二進木の要素のイメージがわかったところで，二進木そのものについて考えてみましょう。実は二進木は先ほどの要素を組み合わせて構成されます。まずは，図9を見てください。各要素のポインタがほかの要素を指し示すことによって，各要素が連結し，ひとつの構造を形作っています。このようなデータ構造を総称して木（tree）といいます。この木のいちばん初めの頭となるもの（厳密にいうとほかのいかなる要素からも指し示されない要素）を根（root）といいます。また，逆にいちばん下の最後についている要素（やっぱり厳密にいうとほかのいかなる要素も指し示さない要素）を葉（leaf）といいます。

ここで気づいてほしいことは，複雑そうなデータ構造も実はそれほど複雑でないということです。この木がきわめて単純な構造をしていることは一目瞭然だと思います。一般にデータ構造の多くは，この木のような単純なモデルの組み合わせとなっています。それはシンプルな構造でデータを扱うほうが効率がよいという現実の結果があるためです（しかしシンプルすぎるのもいけないし，複雑すぎるのもいけない)。したがってデータ構造というものは，そんなに難しいものではありません。

さて，話を二進木に戻しましょう。では，どうしてこの二進木が優れているのでしょうか。それは一般にデータの構造が，データ同士の関係を表現するためです。この二進木の場合，図10のような関係が成り立つようにデータを格納します。つまりある要素から見て，左側の要素に格納された値はすべて自分の値より小さく，右側の要素はすべて自分より大きな値の要素です。したがってある値が木の中にあるかどうか知りたいときは，まず根の値を取り出し値の比較をします。すると，その結果によって調べたい値が右にあるのか左にあるのかがわかります。そこで探したい値があると思われる方向の要素の値を取り出します。この動作を繰り返していけば，調べたい値を簡単に見つけることができます。この様子を図11に示しました。詳しくは，リスト1のinsert関数を見てください。

このような探索を単純に配列でやると，ある値があるかどうかを調べるのに，蓄えられたデータの半分ぐらいの数の比較回数が必要ですし，最初に宣言しただけのデータしか格納できません。しかし，この二進木では配列よりは比較回数は少ないし，あとで述べる動的なデータ構造を使っているので配列のときよりは使いやすくなっています。

insert関数の中にmallocという関数が使われています。この関数は指定したバイト数だけのメモリを主記憶上の未使用領域に確保し（したがってほかのプログラムへの影響はほとんどない），そのアドレスを返します。これをポインタ変数に指し示しておくことで，ユーザーは自由に変数を生成することができます。

このようにプログラムの実行中に変数を生成させたり，消滅させたりすることのできるデータ構造を動的なデータ構造といいます。実はCのおいしさは，この動的なデータ構造にあるといえます。実際Cでは構造体はデータを格納するための器であり，ポインタはそれをくっつける糊であり，関数mallocは箱を切り取ってくる‘はさみ’のようなものです。

したがって，これらをどううまく使うかというのが大事なのです。もっといえば，Cでは構造体とポインタとmalloc関数の三位一体の攻撃（うーん，懐かしい言葉だ。読者にこの言葉からバレーボールを連想できる人がどれだけいるだろうか）こそが，おいしかったりするのです。このおいしさはとてもひと口では語りつくせない，まるで舌の上でとろけるような，それでいてコクがあって……そう，なんとか雄山が腰を抜かし，ミスター味っ子が立ち直れなくなるような，うまさだったりするのでした。そのことから考えると構造体やポインタなどは，それ単体だけではちっともおいしくないし，共用体や記憶クラスなどはこの三位一体の攻撃と比べれば……です。いつの間にか力説してしまいましたが，このことはきわめて重要です。したがって，これらのデータ構造はユーザーの使い方次第では恐ろしいまでの力を発揮します。大事なことはこれらをどのように組み合わせて，いかに効率のよいエレガントなデータ構造を作りあげていくかということであろうと私は思います。

最後にサンプルプログラム内で使われている関数が再帰的になっている点にも注意してほしいなと思います。ここで使った二進木などは自己再帰型と呼ばれるデータ構造です。これは二進木の格好を見ればわかると思いますが，これはデータ構造が自分自身を組み合わせて構成される一種のフラクタル的構造となっているためです。このようなデータ構造について処理をする場合，再帰的プログラミングで処理を行います。これに対し配列などで処理をする場合には，ループで処理をすることになります。で，やっぱり再帰的な処理ができるということも重要なのでした。私にいわせれば再帰ができないC言語なんて，大リーグボールの投げられない星飛雄馬のようなものです。その心は，速いだけでとりえがない。おあとがよろしいようで（うーん，いまひとつだな)。

最後に

Cは低水準な処理ができる言語であるにもかかわらず，きわめて複雑なデータ構造をサポートした言語です。したがってそのプログラムの柔軟性は，ほかの言語には類を見ないものです。逆にいえば，データ構造を十分に利用したプログラムを書けることは，Cを使いこなせるということの条件であると思います。またCの善し悪しにかかわらず，データ構造はプログラミングテクニックのひとつの要素です。これを修得することは，きわめて重要なことであると思います。それだけに，あせらず地道に勉強していきましょう。

第2部 実録Cプログラミング

言語を学ぶ際に重要なことは，文法や書式にとらわれることなく，その言語の本質をつかむことです。C言語の場合，もっとも大事なものはデータ構造の理解でしょう。複雑だからと構造体やポインタを避けて通ることはできません。BASICが初心者のために作られた言語ならC言語はプロのために作られた言語です。表面だけをなでるのではなく，一歩先まで踏み込まないと奥の深さを知ることはできないのです。
以下に挙げられたサンプルプログラムは初心者用としては少し難しいものかもしれません。しかし，どんな言語でも実践を抜きにして会得することはできませんし，C言語をマスターしようとするなら，こういったレベルのプログラムが読めるようにならないと話にならないともいえます。
すでに特集の第1部で基本的な概念についての理解が得られていると思います。その知識をもとにして，これからはソースプログラムがすなわち解説だと思ってください。プログラムにはできるだけ多くの注釈を加え，機種に依存するようなものは一切排除し，さらに標準的なものならば，もっとも低機能なC言語にも対応するように配慮されています。
C言語をお持ちの方なら，入力，そして改造などの順を追っていくのもよいでしょう。できればC言語をお持ちでない方もこの機会にCの考え方に触れてみてください。

迷宮入りの迷路作り

C言語の実践編の一番手にはCビギナーの代表として丹氏の登場です。正しい迷路のあり方とC言語による記述に挑んでもらいました。XCで記述されていますが，基本的にどのCでも動作するはずです。

Tan Akihiko 丹　明彦

正しい迷宮建築とは

人間は昔から迷路というものに不思議な力があると思っていたようです。ギリシャ神話では，怪物ミノタウロスが棲む迷宮あたりが有名だし，世界のあちこちに迷路やそれに似たものが残されています。

現在も迷路と呼ばれるものは世の中にたくさんあります。迷路の本はいうに及ばず，雑誌などのちょっとしたクイズ欄でもちょくちょく見かけます。最近では実際に人が入れるでっかい迷路もいくつか登場してきましたね。こういうものが出てきたことと，パソコン界の「迷宮」――RPGの必須アイテムといってもいいでしょうか――のブームとになんらかのつながりがあることは否定できません。現代人の迷路好きと昨今のRPGブームの間にはなにか関係でもあるのでしょうか。

話を戻しますが，現在（少し前まで，といったほうが正確でしょうが）のパソコン界は「迷宮」であふれかえっています。しかしその大部分は，とても正しい迷路とはいえません。というのも，僕のいう正しい迷路とは，基本的に，

1)　ひとつながりの壁でできている。
2)　「右手法」で脱出が可能。
3)　壁がランダムに配置されている。

というものだからです。「右手法」とは，読んで字のごとく，出発点からずっと右手（別に左手でも足でも構わないのですが）で壁をたどっていく方法のことです。この方法では，いつかは必ず出口にたどりつけますが，かなり無駄な回り道をすることもあります。

先ほど，「基本的に」といったのは，たとえば解く者のウラをかきたい（図1を見てください），とか面白い形の迷路を作りたいとかいった場合にはこの原則を多少崩してもいいと思うからです。そういうわけですから，コンピュータRPGの迷宮は「迷路としては」正しくないといいましたが，決してダメだといっているわけではありません。もし，そうした「正しい」迷路が何階もあって，その中に怪物がウヨウヨいる状態を想像してみてください。こんな迷宮に誰が入りたいと思うでしょうか（「それでも俺は入りたい！」っていう人がどこかにいそうな気がしますね）。

壁を作る竜

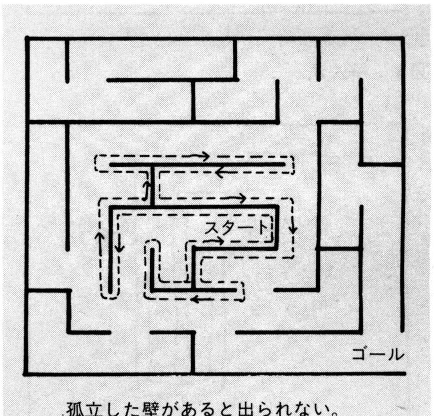
さて，この「正しい」迷路は，意外にもコンピュータを使えば簡単に実現します。単純な規則に従って単調な手順を繰り返せばよいからです。RPGの迷宮などは，その構造に決まった規則がないせいで，コンピュータが自動的に作成することはできず，人間が適当なツールを使って手作業で作ることになります（そこには作る人の個性がある程度現れます。だから，できる迷宮が面白くなる（こともある），といえるのですが）。

で，正しい迷路の作り方の規則を紹介しようと思いますが，RPGの話が出たことだし，ここで1匹の竜（ドラゴン）にご登場を願うことにしましょう。この竜の行動パターンは簡単です。

1)　1歩進むと向きを変えようとする。
2)　しかし決して自分の体にはかみつかないし，自分の体を乗り越えもしない。
3)　胴体は歩くにつれて伸びていく（昔そういうゲームがありましたねえ）。
4)　壁にぶつかると，石と化してしまう（つまり壁の一部となる）。

変なたとえで申しわけありませんが，規則4）あたり，かなりRPGがかっているし（迷宮を作るためだけに作り出されたドラ

図1　右手法の盲点



孤立した壁があると出られない。

ゴンは，用済みになると石にされてしまうのでした！)，少しは親しみを持っていただけたのではないかと思います。

ところで，ここで疑問を感じた方がいるかもしれません。図2のようになったら，この竜はどうするのでしょうか。規則2)があるために，もはや行き場はありません。かといって，壁にぶつかって死ぬこともできないのです。

ここでこの竜に新しい力を与えてやることにしましょう。

5) 行き場がなくなれば，自分の胴体の好きな場所から頭を新しく出して歩き続けられる。

図3を見てください。ちょっと気持ち悪いのですが，これで壁にぶつかるまで心おきなく歩くことができるわけです。

これで僕が竜を引き合いに出して説明した理由がおわかりでしょう。これがたとえば蛇だと，頭がどこからでも出せるなんて芸当はできやしません(竜だったらできるというわけでもないでしょうけど)。

実際に迷路を作るときには，枠を作って，その中に竜の歩幅と同じ間隔で竜の卵を置きます。その卵から順番に竜がかえって歩き回り，壁を作っていくことになるのですが，歩いているうちにほかの卵は竜の胴体の下敷きになってしまいます。こうなった卵は，石の中にいる(！)ので，かえることができないのです(図4)。もしくは，竜は卵を食べながら歩き回ると考えれば(竜の通ったあとには初めから卵がないわけですから)，それでもいいのですが。どちらにしても，壁の中から竜が歩き始めることはありません。

図2 いきづまり

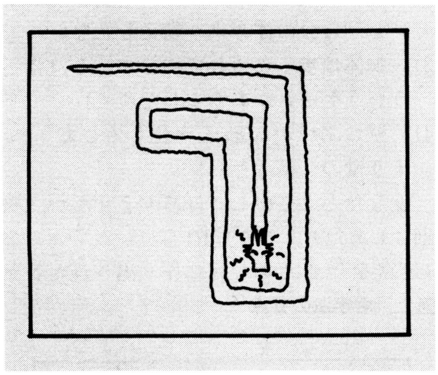
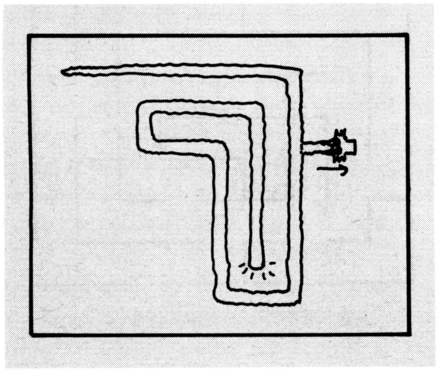


図3 解決法



これだけ知っていれば，紙と鉛筆だけで，どんな大きな迷路だって作れます。

道を掘る竜

さて，迷路を作るには，もうひとつの方法があります。また別の竜に登場してもらいましょう。今度は，石がつまった建物を用意して，その中に竜を放つのです。つまり，この竜は，石の中でないと生きていけないのです。石の中をでたらめに掘り進むだけが商売なのです。そして空気に触れたときに，あとかたもなく消えてしまうのです。あとの行動パターンは同じです。ただし，枠から外に出ることはできません。要するに，前の竜は壁を作りましたが，今度の竜は道を作るのです。

卵を置く場所は少し違います(図5)。それに，入り口を1カ所ちゃんと開けておかないと，この竜は空気に触れない限り死ねないので，歩き回るうちに迷路いっぱいになってしまい，動けなくなります(図6)。この段階で，道はすでにできあがっているのですが，竜が死にきれないために，迷路が完成しないのです。

ところで，この方法を使えばもう少し面白いことができるのです。今までの竜は前進と左折または右折しかできませんでしたが，さらに上または下に動けるようにしてやるのです。するとどうなるでしょう。立体迷路，つまり2階建て，3階建ての迷路ができあがるのです。といっても，RPGに出てくるような何階建てという迷宮よりもはるかに難しいものです。上や下に行く階段が，いたるところにあるようなものですから。ただしRPGの迷宮には，魔法がかかっているために解けないというものが多いので，いくら立体迷路が難しいといっても，ただ複雑だから難しいというだけです。

図4 石の中にいる

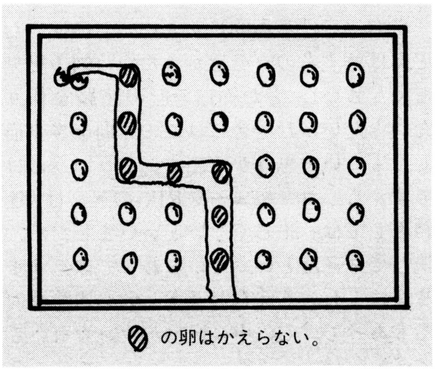


図5 卵を置く場所

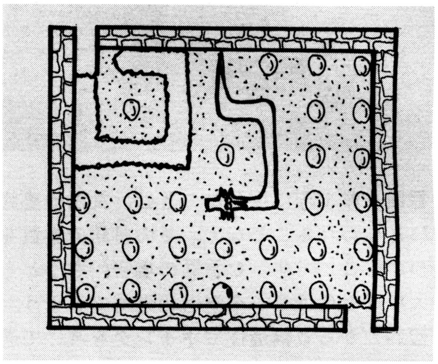
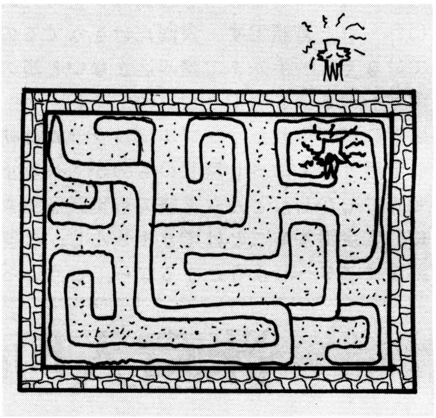


図6 竜が死にきれない



なお，立体迷路は道を掘り抜く竜にしか作れません。壁を作る竜が同じことをやっても，おそらく迷路の形にならないと思います。

迷路を解く竜

作った迷路は解かれねばなりません。そこで解き方を。平面の迷路だったら右手法が早いでしょう。きれいな答え(余分な回り道のないもの)がほしければ，右手法と左手法で別々に解いて，両方の道が重なったところを取り出せば1本道の答えが得られます。しかし，立体迷路には右手法は通用しません。そこで，ここではどの方法で作った迷路にも使えるやり方を紹介しておきましょう。第3の竜が登場します。この竜は少し変わっています(今までのだって変わっていましたが)。

1) 行き止まりに置かれた卵から生まれる。
2) 1本道しか通れない。例によって体を伸ばしながら歩く。
3) 分かれ道にくると死んでしまう。死骸はその場に残る。
4) これを繰り返すと，最後には道が1本だけ残る。これが答え。

図7に考え方を示しておきます。

実践編 さあプログラミング

少し話が長くなってしまいましたが，これらをCでプログラムしてみましょう。どのCでも基本は同じことですが，ここではXCを使っています。

リスト1は壁をのばす竜のプログラムmaze1.cです。特に変わったことはしていません。ヘッダ（要するにmain( )の前）がムヤミと長いので，BASICあがりのプログラマには多少うっとうしく感じられるかもしれませんが，おまじないだと思って打ち込んでください。

プリプロセッサは，うまく使えば「ちょっとここを変えたいな」などというときは極楽です。たとえば17行目の，

```
#define    wall    'ロ'
```

というところです。これは壁のキャラクタですが，ほかのキャラクタにしたいならここを変えるだけですみます。リストの中を探し回る必要はありません。

あと，BASICあがりの人が陥りやすいミスは，演算子だとか，カッコ（制御構造の{ }や，配列の添字の[ ]など）の数の間違いだとかいったものですが，これらはすべてコンパイラのほうでエラーを出してくれます。Cで本当に恐ろしいのは，コンパイラにひっかからないエラー（こういうのは「バグ」というのでしょうが，なかには「エラー」としてコンパイラに検出してほしいものもあります）で，ここでBASICあがりは挫折し，BASICの面倒見のよさ，そのありがたみを知ることになるのです。まあ，脅かしていないで先に進みましょう。

構造体を使う

リスト2は道を掘る竜のプログラムmaze2.cですが，maze1.cと少し違います。Cは変数の型の定義ができますので，vector（ベクトル）という型を，構造体を使って定義しています（23行）。

XCのリファレンスマニュアルによると，構造体は，「“型の異なる”複数の変数の集合をひとつのデータ単位として扱うもの」となっています。このvectorという型は，座標をひとまとめにして扱うために作ったもので，メンバも“同じ型の”整数3つだけです。事実，この程度なら配列とポインタをうまく使えば構造体など要らないともいえるのですが，ベクトルというものの性質上，これをさらに配列にしたり，代入したりといったことが簡単にできないと非常に困るのです。このような目的には構造体はうってつけです（似たような例に複素数があります。Cのライブラリでは，複素数を構造体を使って定義していますが，メンバは同じ実数型変数2つだけです）。

少しうっとうしい話になってしまいましたが，わざわざこう定義したメリットがどういうところに現れているか見てみましょう。まず，プログラムの主要部分からxとかyとかいった座標変数が消えています。ベクトルは，座標をひとまとめにして数と同じように扱えるようにするためのものですから，xやyを別々に処理することなくプログラムが書けるのです。もっとも，ベクトル自体の演算などの低レベルな処理では座標変数も必要ですが。

ところで，どうしてzなどというものがあるのでしょうか。実は，ここにベクトルを採用した最大のメリットがあるのです。いったんベクトルにすると，2次元だろうが3次元だろうが（4次元だってできますが，そうなるともはや人間には解けません。仮に作ったとして，zの次の座標はt，つまり時間なのでしょうか。刻々と形を変える迷路なんてあったら面白いでしょうが，僕はそんな迷路には入りたくありません）そのたびにわざわざプログラムを変える必要はなくなってしまうのです。つまり，こうすると平面だけでなく立体の迷路も作れるのです。具体的にはどうすればいいでしょう。変更点はたったの2カ所。プリプロセッサに指定するところで，

dimensionを2から3に

size_zを3からもっと大きな奇数に

するだけでよいのです（迷路のサイズは，道と壁の幅が同じになるようにプログラムしてある関係上，必ず奇数にしなくてはなりません）。もし，x, y, zを別々に処理していると，プログラムにかなりの変更を加える必要があったでしょう。

第一の障害

まあ，とにかくリスト1の座標変数をベクトルで書き換えて，さらに道を掘り抜くように変更したプログラムを作ってみました。直ちにコンパイル，そして実行。

いきなり，グチャグチャに壊れた迷路が出てきました。なぜなのでしょうか。もしかしたら，ベクトルなどという型を勝手に作ったので神の怒りが下ったのでしょうか。この原因は，BASICで同じようなプログラムを作ってみるとわかります（もちろんベクトルは使えませんが）。BASICインタプリタは次のようなメッセージを出してくれるはずです。

添字の値が異常です

結論からいいますと，このとき竜は迷路の建物からはみ出してしまったのです。壁を作る竜の場合は端っこの壁のところで止められたのでエラーは出なかったのですが，道を掘る竜ではそうはいかず（壁となる石と，枠とは区別されていないのです），迷路の外（つまり配列の定義されていない部分）までめくらめっぽうに食い荒らそうとしたために，妙な迷路ができあがってしまったのです。Cのオブジェクトコードには，配列の添字のチェックなどといった，実行速度を落とすようなものはいっさい入っていないので，ここから致命的なエラーを起こすことがしばしばあります。この場合は，ヘタをすると竜がプログラム本体を壊してしまう可能性もあったわけです。ああ恐ろしい。

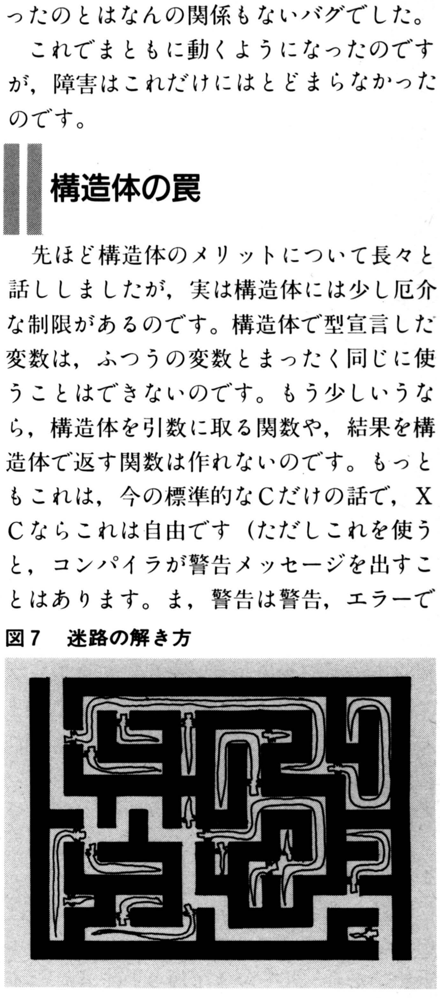
というわけで，ここのデバッグをした結果がoutcheck( )というルーチンです（129行）。竜が迷路からはみ出そうとすると，強制的に動きを止めさせます。要するに，これはアルゴリズムのミスで，ベクトルを使ったのとはなんの関係もないバグでした。

これでまともに動くようになったのですが，障害はこれだけにはとどまらなかったのです。

構造体の罠

先ほど構造体のメリットについて長々と話ししましたが，実は構造体には少し厄介な制限があるのです。構造体で型宣言した変数は，ふつうの変数とまったく同じに使うことはできないのです。もう少しいうなら，構造体を引数に取る関数や，結果を構造体で返す関数は作れないのです。もっともこれは，今の標準的なCだけの話で，XCならこれは自由です（ただしこれを使うと，コンパイラが警告メッセージを出すことはあります。ま，警告は警告，エラーで

図7　迷路の解き方



はないので無視して実行させると，ちゃんと動きますけど)。将来はこんな制限もなくなるのでしょうけど(？)。

実は，僕はXCで作っていて，この制限を知らずにプログラムを組み上げてしまっていたのを，「それはいけないよ」と指摘されて，あとから変更するハメになったのでした。「全部ポインタを使えばいいじゃないか」……そう，文字列のように，ポインタを利用してしまえばいいのです。

まず僕がやった変更は，小さなサブルーチンの中で構造体のメンバを参照しているところでした。ふつうの構造体のメンバ参照式は「．」(ピリオド)ですが，ポインタから参照する場合は「－＞」ですから，これだけを機械的に変えていきました。次に，関数はポインタで呼び出すんだからと，引数になっているベクトル変数の名前の頭に間接演算子（＊）をくっつけて回りました。まあこんなもんだろうと思ってさっそくコンパイルしてみると，

　間接演算子（＊）がポインタ以外に使用されている

というエラーが出てしまいました。

どうも僕は間接演算子の意味を取り違えていたようです。確かに「＊」をつけたベクトルは，ベクトルという値であって，ポインタではありません。関数を呼び出す側では，「＊」ではなくアドレス演算子（&）を使うのが筋でしょう。「&」は変数の「格納アドレス」を与え，「＊」はポインタが指す「アドレスの内容」を参照するのですから。ううむややこしい。

ポインタの迷路を抜けて

というわけですから，呼び出す側では「&」，受け取る関数の側では「＊」，というふうにすればいいのでしょう。今度は無事コンパイルが終了しましたから，意気揚々と実行させたのです。

　バスエラーが発生しました

おっとっと。いきなりこの攻撃には参りました。コンパイラを通ったプログラムが，マシン語レベルのエラーを出してストップするとは！　幸いX68000で作業していましたからエラーで中断できましたが，これがZ80だったら暴走しています。いったいどうしたというのでしょうか。これは，コンパイルエラーが出ていないだけになかなかの難問です。

関数mazemake( )中のHEAD(83行)は引数のベクトルではありますが，メインルーチンから「&」つきで渡されてきたポインタなのです。僕は，このHEADにもう一度「&」をつけてほかの関数を呼び出すのに使ってしまっていました。すると，その関数にはポインタが格納されているアドレスが渡り，呼び出された関数はそれをポインタと思い込んで受け取り，そこで構造体のメンバ，つまりベクトルの座標値を参照するつもりが，ポインタの値自体を読み出してしまい，座標値を使って迷路の要素を読み出すつもりが，変なアドレス（おそらくスーパーバイザ領域でしょう。もしかするとメモリが実装されていない領域かも）から読み出そうとしてしまい……，というのが真相なのでしょう。

なんだかわけがわからなくなってしまいました（ポインタをきちんと理解している人ならなんとかわかるんじゃないでしょうか）が，要するに，HEADの前の「&」が悪人なのです。では，どうすればいいのでしょうか。困って，関数mazemake( )の頭の引数定義の部分を見ると，

　vector　＊HEAD;

とあるではありませんか。「ああそうか，この関数の中ではHEADは『＊』つきでやっと一人前なんだな」と思った僕は，いそいそとHEADの前の「&」を「＊」へと変えたのです。そこでコンパイル。

　関数呼び出しでの引数の型が正しくない

　構造体/共用体が関数への引数として指定されている

などとコンパイラは警告メッセージを出しましたが，オブジェクトはちゃんと出てきたので，「エラーじゃないし，まあいいや。構造体に警告はつきものなんだろう」とばかりに実行させたのです。

　アドレスエラーが発生しました

むむむいったいぜんたいどういうことなのでしょう。もうお手上げだあ！

解決編

……などと悩んだふりをするのはやめて，さっさと種明かしをしてしまいましょう。なんのことはありません。実はHEADにはなにもつけなくていいのです。mazemake( )をもう一度よく見ましょう。関数宣言部の引数リスト，つまりカッコ内には

　vector mazemake(HEAD)

と，HEADが「＊」なしで書いてあります。これこそがポインタです。「＊」をつけると，そのアドレスの内容，すなわちベクト

リスト3　バスエラー修正前

```
void     mazemake( HEAD )
vector   *HEAD;
{
         vector            NEWHEAD, HEADMOVE;
         int               i, r, l;
         unsigned char     c;

         length=0;
         mazeput( *HEAD, body );                                     /*vectorの値を*/
         while ( mazeget( *HEAD )!=air ) {                           /*  引数にしては*/
                 v_cpy( &DRAGON1[length], *HEAD );                   /*  いけません*/
                 for ( i=0; i<=(dimension*2-1); i++ ) {              /*  *HEADが間違い*/
                         r=random(dimension*2)+1;
                         v_mul( &HEADMOVE, &DIRECTION[r], 2 );
                         v_add( &NEWHEAD, *HEAD, &HEADMOVE );
                         if ( outcheck( &NEWHEAD ) ) {
                                 v_cpy( &NEWHEAD, *HEAD );
                                 r=0;
```

リスト4　アドレスエラー修正前

```
void     mazemake( HEAD )
vector   *HEAD;
{
         vector            NEWHEAD, HEADMOVE;
         int               i, r, l;
         unsigned char     c;

         length=0;
         mazeput( &HEAD, body );                                     /*ポインタへのポインタなんて*/
         while ( mazeget( &HEAD )!=air ) {                           /*  意味がありません*/
                 v_cpy( &DRAGON1[length], &HEAD );                   /*  &HEADが間違い*/
                 for ( i=0; i<=(dimension*2-1); i++ ) {
                         r=random(dimension*2)+1;
                         v_mul( &HEADMOVE, &DIRECTION[r], 2 );
                         v_add( &NEWHEAD, &HEAD, &HEADMOVE );
                         if ( outcheck( &NEWHEAD ) ) {
                                 v_cpy( &NEWHEAD, &HEAD );
                                 r=0;
```

ルの値を指すのです。引数の型宣言をしているところの「vector *HEAD」は，「この関数はベクトルへのポインタを引数としてとるんだけど，vectorという型の宣言だから（ポインタじゃないから）『*』をつけて，値のかたちで定義してるんだよ」ということなのでしょう。

この関数の中からさらにほかの関数を呼び出そうとするなら，このプログラム中の関数の引数はポインタでなくてはなりませんので，なにもつけないHEADで呼び出すのです。ちなみに，「&」をつけるのは，一応ポインタですから文法的には間違っていないので，エラーも警告も出ずにコンパイルは終了しますが，このポインタはベクトル変数へのポインタではなく，ベクトル変数へのポインタへのポインタですから（あー面倒臭い），とんでもないアドレスを指します。この場合，ポインタへのポインタなんてまったくのナンセンスといえるでしょう（こういう特殊な使い方ができないとはいいきれませんが）。

ちょっと整理しましょう。要するに次のように書いてくれればいいのです。

```
void    main( )
{
        vector   V;            ：値
          ⋮
        func1(&V);             ：ポインタ
          ⋮
}
void    func1(V1)              ：ポインタ
vector  *V1;                   ：内容(値)
{
        int      a;
        vector   V2;           ：値
          ⋮
        a=func2(V1,&V2);       ：ポインタ
          ⋮                    （どちらも）
}
int     func2(Va,Vb)           ：ポインタ
vector  *Va, *Vb;              ：内容(値)
{
        int      x;
          ⋮
        return(x);
}
```

あまりうまく書けませんでしたが，感じくらいはつかめるでしょう。

ともかく，これでめでたく動くようになりました。でも，リストを見ると，関数の引数に&がついたりつかなかったりしているし，関数の呼び出しを見ただけでは，引数がポインタなのか値なのか，全然わからないこともあるのです。これははっきりいって見苦しいと思います。早くすべてのCが，構造体を制限なしで扱えるようになってほしいと願う次第でした。

このように，Cという言語は，変数のチェックが非常に甘い（その代わりに強力であるともいえるのですが）ので，よく暴走を起こします。バスエラーやアドレスエラーは，ランタイムルーチンが常に監視していて変数をチェックした結果がおかしかったからというのではなく，あくまでオブジェクトプログラムがシステムに対して悪事を働こうとしたために起きたエラーです。いくら実行が止まったからといって，両者を混同してはいけません。これはシステムがストップをかけたにすぎず，MC68000の力によるものなのですから。Cはそれほど親切ではありませんよ。

最後は迷路を解くプログラムsolve.cです。このうちのいくつかの関数はmaze2.cと共通のものです。このプログラムでは，ファイルの扱いが少々面倒でしたが，Cならでは，というエラーも出なかったので，デバッグのお話は省略します。Cはコマンドラインからのパラメータを簡単に受け取れますのでファイル処理にはうってつけだといえるでしょう。

プログラムの使い方はどれも同じです。コマンドラインから

```
maze1  〈filename〉
maze2  〈filename〉
maze3  〈filename〉
solve  〈filename〉
```

とやります。maze1とmaze2は〈filename〉で指定されたファイルに迷路を出力します。スクリーンエディタにでもかけて遊んでください。プリンタに打ち出すのもいいでしょう。ちなみにmaze2に前にいった2カ所の変更をして3次元仕様に直してやると，「どうやって解けというんだ!?」といいたくなるような出力をします。それからsolveというのは，〈filename〉で与えられた迷路の不要な道を塗りつぶして正解を教えてくれます。

最後に

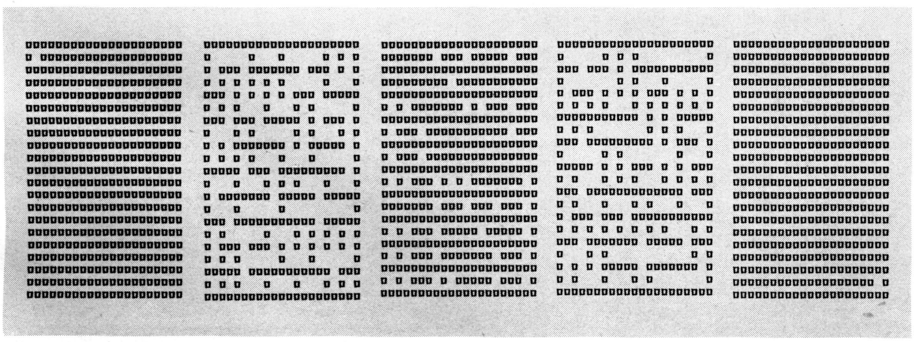
本当は3次元迷路と称して，作った迷路の中に入りたかったのですが，それはRPGでやり飽きてるだろうし，そもそもプログラムが大きくなりすぎるからやめにします。暇な方は，メモリの続く限り大きい迷路を作ってみてはいかがでしょうか。そのときはプリプロセッサのsize_x, size_y, size_z（それぞれ迷路のサイズ。すでにいったように必ず奇数にすること），maxlength（1匹の竜の体長。いくら大きくしても，配列がメモリを食うだけで，プログラムの実行に影響はありません）の値を大きくしてください。ただし，このときは，solve.cのプリプロセッサの値も大きくしないと，解けない迷路ができてしまいます。

ついでにいいますが，乱数はいつも同じなので，迷路がワンパターンになってしまいます。変えたいときは，randomizeの値を変えてください。

できればポインタは避けて通りたかったのですが，さすがにそういうわけにもいかなかったようです。Cプログラマになろうと思ったら，ポインタぐらいは使えなくてはいけないのでしょうが，こいつはなんともイヤラシいシロモノでした。皆さんも心してかかってください。システムを2，3回破壊するくらいの覚悟は必要です（特にZ80の人）。

以上，XCでの開発を例にとって，いろいろとCプログラマのひっかかるワナにはまってみせましたが（わざとひっかかったものもあるし，本当にひっかかって悩まされたものもありますが），このプログラムはZ80のCでも動くはずです。くれぐれも暴走には気をつけましょう。もっとも暴走させた経験からはいろいろなことが学べますから，悪いことばかりでもありません（そんなことないか）。それではこれにてさようなら。

図8　立体迷路の出力例



リスト1 maze1.c

```
/*
        maze making program
                dragon method ( grow & stone )
*/


#include        <stdio.h>                                       /*インクルードファイル*/
#include        <stdlib.h>


#define         size_x          21                              /*迷路の大きさ*/
#define         size_y          21                              /*  （奇数に限る）*/
#define         maxlength       256                             /*竜の体長の最大値*/
#define         randomize       1                               /*乱数系列*/

#define         air             ' '                             /*空間（道）*/
#define         wall            '□'                             /*壁*/
#define         body            '#'                             /*竜の胴体*/


unsigned char   maze[size_x][size_y];                           /*迷路本体*/
int             dragon1_x[maxlength], dragon1_y[maxlength],     /*竜の胴体の座標（その1）*/
                dragon2_x[maxlength], dragon2_y[maxlength],     /*竜の胴体の座標（その2）*/
                dx[4]={-1, 1, 0, 0}, dy[4]={0, 0, -1, 1};       /*竜の頭の変位*/
int             length;                                         /*竜の体長*/


void    mazemake();
void    stone();
void    mazeclr();                                              /*関数のフォワード宣言*/
void    makewall();
void    makedoor();
void    printout();
void    output();
int     random( int );



void    main( argc, argv )
int     argc;                                                   /*コマンドラインからの*/
char    *argv[];                                                /*      パラメータ*/
{
        int     x0, y0;

        mazeclr();                                              /*迷路を初期化*/
        makewall();                                             /*迷路の枠を作る*/
        srand(randomize);                                       /*乱数系列を初期化*/
        for ( y0=2; y0<=size_y-3; y0+=2 ) {                     /*卵を置いていく*/
                for ( x0=2; x0<=size_x-3; x0+=2 )               /*  壁の下なら死んでいる*/
                        if ( maze[x0][y0]!=wall )               /*  そうでないなら*/
                                mazemake( x0, y0 );             /*  竜は歩き始めるので*/
        }                                                       /*  迷路の壁を作らせる*/
        makedoor();                                             /*出入口をあける*/
        printout();                                             /*画面に出力する*/
        if ( argc>1 ) output( argv[1] );                        /*ファイル名の指定があれば*/
}                                                               /*  そちらにも出力する*/


void    mazemake( x, y )                                        /*(x,y)に置かれた*/
int     x, y;                                                   /*  卵からかえった*/
{                                                               /*  竜が壁を作る*/
        int             x1, y1, x2, y2, i, r, l;
        unsigned char   c;

        length=0;
        maze[x][y]=body;                                        /*出発点に頭を置く*/
        while ( maze[x][y]!=wall ) {                            /*石になるまで*/
                dragon1_x[length]=x;                            /*頭の座標*/
                dragon1_y[length]=y;
                for ( i=0; i<4; i++ ) {                         /*動く向きを決める*/
                        r=random(4);                            /*  4方向から選択*/
                        x1=x+dx[r]*2;                           /*頭の行き先を探る*/
                        y1=y+dy[r]*2;                           /*なるべく胴体に*/
                        c=maze[x1][y1];                         /*  噛みつかないように*/
                        if ( c!=body ) break;                   /*  4回まで調べる*/
                }
                switch ( c ) {                                  /*行き先が*/
                        case air:                               /*空間だったら胴体を伸ばす*/
                                x2=dragon2_x[length]=x+dx[r];
                                y2=dragon2_y[length]=y+dy[r];
                                maze[x2][y2]=body;              /*新しい首を置く*/
                                maze[x1][y1]=body;              /*新しい頭を置く*/
                                x=x1; y=y1;
                                length++;                       /*胴体を伸ばす*/
                                break;
                        case wall:                              /*壁だったら石になる*/
                                x2=dragon2_x[length]=x+dx[r];
                                y2=dragon2_y[length]=y+dy[r];
                                maze[x2][y2]=body;
                                x=x1; y=y1;
                                stone();
                                break;
                        case body:                              /*胴体だったら*/
                                dragon2_x[length]=x;            /*  頭を別のところから出す*/
                                dragon2_y[length]=y;
                                l=random(length);
                                x=dragon1_x[l];
                                y=dragon1_y[l];
                                length++;
                                break;
                }
        }
}
```

▶うちのX68000は凄い。BGMは流れるわ，30秒間隔で画面の色が変わるわでもう大変！このごろはパーサと格闘している毎日です。　木村　昌樹（16）山形県

```
void    stone()                                              /*竜を石にする*/

{
        int     i;
        for ( i=0; i<=length; i++ ) {
                maze[dragon1_x[i]][dragon1_y[i]]=wall;
                maze[dragon2_x[i]][dragon2_y[i]]=wall;
        }
}


void    mazeclr()                                            /*迷路を初期化する*/
{
        int     x, y;
        for ( y=0; y<size_y; y++ ) {
                for ( x=0; x<size_x; x++ ) {
                        maze[x][y]=air;
                }
        }
}


void    makewall()                                           /*迷路の枠を作る*/
{
        int     x, y;
        for ( y=0; y<size_y; y++ ) {
                maze[0][y]=wall;
                maze[size_x-1][y]=wall;
        }
        for ( x=0; x<size_x; x++ ) {
                maze[x][0]=wall;
                maze[x][size_y-1]=wall;
        }
}


void    makedoor()                                           /*出入り口をあける*/
{
        maze[1][0]=air;
        maze[size_x-2][size_y-1]=air;
}


void    printout()                                           /*画面に出力する*/
{
        int     x, y;
        for ( y=0; y<size_y; y++ ) {
                for ( x=0; x<size_x; x++ )
                        printf( "%c", maze[x][y] );
                printf( "¥n" );
        }
}


void    output( filename )                                   /*ファイルに出力する*/
char    *filename;                                           /*ファイルネームは*/
{                                                            /*  ポインタで受け取る*/
        int     x, y;
        FILE    *mazefile;
        mazefile=fopen( filename, "w" );
        for ( y=0; y<size_y; y++ ) {
                for ( x=0; x<size_x; x++ )
                        fprintf( mazefile, "%c", maze[x][y] );
                fprintf( mazefile, "¥n" );
        }
        fclose( mazefile );
}


int     random( n )                                          /*0～n-1の乱数を返す*/
int     n;
{
        int     r;

        r=rand() / (32768/n);

        return( r );
}
```

リスト2 maze2.c

```
/*
        maze making program
                dragon method ( dig & lost )
*/


#include        <stdio.h>
#include        <stdlib.h>

#define         dimension       2                     /*立体迷路では3に変える*/

#define         size_x          21                    /*迷路の大きさ*/
#define         size_y          21                    /*  (奇数に限る)*/
#define         size_z          3                     /*立体迷路なら5以上の奇数*/
#define         maxlength       1024
#define         randomize       1

#define         air             0
#define         wall            255
#define         body            1
```

▶『理工系のBASIC入門』とはなんという本だ！　著者が全員青山学院大学の先生ではないか。これではまるで青山の教科書のために書かれたみたいじゃないか。しかも著者のひとりは私の先生ではないか。
泉　昭彦（18）神奈川県

```
typedef struct { int a[3]; }    vector;                         /*型vectorの定義*/

unsigned char   maze[size_x][size_y][size_z];                   /*迷路本体*/
vector          DRAGON1[maxlength],                             /*竜の胴体の座標*/
                DRAGON2[maxlength],
                DIRECTION[7]={                                  /*竜の頭の変位*/
                         0, 0, 0,                               /*構造体の配列の*/
                        -1, 0, 0,                               /*  初期化のしかた*/
                         1, 0, 0,
                         0,-1, 0,
                         0, 1, 0,
                         0, 0,-1,
                         0, 0, 1};
int             length;


void            mazemake( vector * );                           /*vector型の引数は*/
int             outcheck( vector * );                           /*  すべてポインタ*/
void            lost();                                         /*  になっている*/
void            mazefill();
void            makedoor();
void            printout();
void            output( char * );
int             random( int );
void            v_cpy( vector *, vector * );
void            v_add( vector *, vector *, vector * );
void            v_mul( vector *, vector *, int );
unsigned char   mazeget( vector * );
void            mazeput( vector *, unsigned char );



void    main( argc, argv )
int     argc;
char    *argv[];
{
        int     x0, y0, z0;
        vector  EGG;

        mazefill();                                             /*迷路を初期化する*/
        makedoor();
        srand(randomize);
        for ( z0=1; z0<=size_z-2; z0+=2 ) {
                for ( y0=1; y0<=size_y-2; y0+=2 ) {
                        for ( x0=1; x0<=size_x-2; x0+=2 ) {     /*卵を置いていく*/
                                if ( maze[x0][y0][z0]==wall ) { /*  そこが壁だったら*/
                                        EGG.a[0]=x0;            /*  卵はかえるので*/
                                        EGG.a[1]=y0;            /*  竜に道を掘らせる*/
                                        EGG.a[2]=z0;
                                        mazemake( &EGG );
                                }
                        }
                }
        }
        printout();
        if ( argc>1 ) output( argv[1] );
}


void    mazemake( HEAD )                                        /*HEADに置かれた*/
vector  *HEAD;                                                  /*  卵からかえった*/
{                                                               /*  竜が道を掘る*/
        vector          NEWHEAD, HEADMOVE;
        int             i, r, l;                                /*この関数では、HEADと*/
        unsigned char   c;                                      /*  &NEWHEAD, &HEADMOVEという*/
                                                                /*  使い方の差に注意する*/
        length=0;
        mazeput( HEAD, body );                                  /*出発点に頭を置く*/
        while ( mazeget( HEAD )!=air ) {                        /*消え去るまで*/
                v_cpy( &DRAGON1[length], HEAD );                /*頭の座標*/
                for ( i=0; i<=(dimension*2-1); i++ ) {          /*動く向きを決める*/
                        r=random(dimension*2)+1;                /*  2次元なら4方向*/
                        v_mul( &HEADMOVE, &DIRECTION[r], 2 );   /*  3次元なら6方向*/
                        v_add( &NEWHEAD, HEAD, &HEADMOVE );     /*  から選択*/
                        if ( outcheck( &NEWHEAD ) ) {           /*行き先を採る*/
                                v_cpy( &NEWHEAD, HEAD );        /*なるべく胴体に*/
                                r=0;                            /*  噛みつかないように*/
                        }                                       /*  2次元なら4回*/
                        c=mazeget( &NEWHEAD );                  /*  3次元なら6回*/
                        if ( c!=body ) break;                   /*  まで調べる*/
                }
                switch ( c ) {                                  /*行き先が*/
                        case wall:                              /*壁なら掘り進む*/
                                v_add( &DRAGON2[length], HEAD, &DIRECTION[r] );
                                mazeput( &DRAGON2[length], body );
                                mazeput( &NEWHEAD, body );
                                v_cpy( HEAD, &NEWHEAD );
                                length++;
                                break;
                        case air:                               /*空間なら消え去る*/
                                v_add( &DRAGON2[length], HEAD, &DIRECTION[r] );
                                mazeput( &DRAGON2[length], body );
                                v_cpy( HEAD, &NEWHEAD );
                                lost();
                                break;
                        case body:                              /*胴体なら頭を別に出す*/
                                v_cpy( &DRAGON2[length], HEAD );
                                l=random(length);
                                v_cpy( HEAD, &DRAGON1[l] );
                                length++;
                                break;
                }
        }
}
```


▶最近残業で帰ると、妻が源平討魔伝をやっています。やるヒマのない私に妻は「私が先にクリアする」と燃えています。上海でもかなわなかったし、マンハッタンレクイエムも解かれてしまったばかり。このままではX68000はどうなるのだろうと心配の毎日です。
高橋　智行（26）埼玉県

```
int     outcheck( V )                                              /*竜が外に出るのを防ぐ*/
vector  *V;                                                        /*引数はポインタで*/
{
        int     x, y, z, out=0;

        x=V->a[0];                                                 /*ポインタからのメンバ参照*/
        y=V->a[1];
        z=V->a[2];
        if ( x<=0 || x>=size_x || y<=0 || y>=size_y || z<=0 || z>=size_z ) out=1;

        return( out );
}


void    lost()                                                     /*竜を抹殺する*/
{
        int     i;

        for ( i=0; i<=length; i++ ) {
                mazeput( &DRAGON1[i], air );
                mazeput( &DRAGON2[i], air );
        }
}


void    mazefill()                                                 /*迷路を壁で埋める*/
{
        int     x, y, z;

        for ( z=0; z<size_z; z++ ) {
                for ( y=0; y<size_y; y++ ) {
                        for ( x=0; x<size_x; x++ ) {
                                maze[x][y][z]=wall;
                        }
                }
        }
}


void    makedoor()
{
        int     z;

        for ( z=0; z<=1; z++ )
                maze[1][1][z]=air;
        maze[size_x-2][size_y-2][size_z-1]=air;
}


void    printout()
{
        int     x, y, z;

        for ( z=0; z<size_z; z++ ) {
                for ( y=0; y<size_y; y++ ) {
                        for ( x=0; x<size_x; x++ ) {
                                switch ( maze[x][y][z] ) {
                                        case air:
                                                printf( " " ); break;
                                        case body:
                                                printf( "#" ); break;
                                        case wall:
                                                printf( "□" ); break;
                                }
                        }
                        printf( "¥n" );
                }
                printf( "¥n" );
        }
}


void    output( filename )
char    *filename;
{
        int     x, y, z;
        FILE    *mazefile;

        mazefile=fopen( filename, "w" );
        for ( z=0; z<size_z; z++ ) {
                for ( y=0; y<size_y; y++ ) {
                        for ( x=0; x<size_x; x++ ) {
                                switch ( maze[x][y][z] ) {
                                        case air:
                                                fprintf( mazefile, " " ); break;
                                        case wall:
                                                fprintf( mazefile, "□" ); break;
                                }
                        }
                        fprintf( mazefile, "¥n" );
                }
                fprintf( mazefile, "¥n" );
        }
        fclose( mazefile );
}


int     random( n )
int     n;
{
        int     r;

        r=rand() / (32768/n);

        return( r );
}
```

▶皆さん，もう知っていると思いますが源平討魔伝の源平main.xなどのファイルをダンプしてみると面白いメッセージが読めますよ。SHELL＝源平討魔伝.Xも笑えます。
荒井　和彦 (28) 群馬県

```
void    v_cpy( V1, V2 )                                     /*ベクトルのコピー*/
vector  *V1, *V2;
{
        int     i;

        for ( i=0; i<3; i++ )
                V1->a[i]=V2->a[i];
}


void    v_add( V1, V2, V3 )                                 /*ベクトルの加算*/
vector  *V1, *V2, *V3;
{
        int     i;

        for ( i=0; i<3; i++ )
                V1->a[i]=V2->a[i]+V3->a[i];
}


void    v_mul( V1, V2, t )                                  /*ベクトルの整数倍*/
vector  *V1, *V2;
int     t;
{
        int     i;

        for ( i=0; i<3; i++ )
                V1->a[i]=V2->a[i]*t;
}


unsigned char   mazeget( V )                                /*ベクトルで示される場所の*/
vector  *V;                                                 /*  状態を調べる*/
{
        int     x, y, z;

        x=V->a[0];
        y=V->a[1];
        z=V->a[2];

        return( maze[x][y][z] );
}


void    mazeput( V, c )                                     /*ベクトルで示される場所に*/
vector          *V;                                         /*  キャラクタを入れる*/
unsigned char   c;
{
        int     x, y, z;

        x=V->a[0];
        y=V->a[1];
        z=V->a[2];
        maze[x][y][z]=c;
}
```

リスト5 solve.c

```
/*
        maze solving program
                dragon method ( grow & dead )
*/


#include        <stdio.h>
#include        <stdlib.h>

#define         maxsize_x       127                         /*大きな迷路でも読み込める*/
#define         maxsize_y       127
#define         maxsize_z       9

#define         air             0
#define         wall            255
#define         body            1


typedef struct { int a[3]; }    vector;

unsigned char   maze[maxsize_x][maxsize_y][maxsize_z];
vector          DIRECTION[7]={
                        0, 0, 0,
                       -1, 0, 0,
                        1, 0, 0,
                        0,-1, 0,
                        0, 1, 0,
                        0, 0,-1,
                        0, 0, 1};
int             dimension=3, size_x, size_y, size_z;


void            mazesolve( vector * );
int             status( vector * );
void            printout();
void            input( char * );
void            output( char * );
void            mazecopy( int, int );
void            v_add( vector *, vector *, vector * );
void            v_mul( vector *, vector *, int );
unsigned char   mazeget( vector * );
void            mazeput( vector *, unsigned char );

```

▶6月号の77ページで「ナインという線もありますが」といっている編集の人，あなたは立派な“おたく”だと思います。ジャンボーグAなんて，もう誰も覚えていませんよ。編集後記からしてUさんですか……。 斉藤 典昭 (21) 神奈川県

```
void    main( argc, argv )
int     argc;
char    *argv[];
{
        int     x0, y0, z0;
        vector  EGG;

        if ( argc>1 ) {                                         /*ファイル名の指定は必要*/
                input( argv[1] );                               /*  それを読み込む*/
                for ( z0=1; z0<=size_z-2; z0+=2 ) {
                        for ( y0=1; y0<=size_y-2; y0+=2 ) {     /*卵を置いていく*/
                                for ( x0=1; x0<=size_x-2; x0+=2 ) {
                                        EGG.a[0]=x0;            /*いちおう全部かえる*/
                                        EGG.a[1]=y0;
                                        EGG.a[2]=z0;
                                        mazesolve( &EGG );      /*行き止まりを埋めさせる*/
                                }
                        }
                }
                if ( dimension==2 ) {                           /*maze1.cだけ*/
                        mazecopy(0,1);                          /*  データ構造が違うので*/
                        size_z=1;                               /*  その対策*/
                }
                printout();
                output( argv[1] );
        }
}


void    mazesolve( HEAD )                                       /*迷路の行き止まりを埋める*/
vector  *HEAD;
{
        int     s;

        while( (s=status(HEAD))!=0 ) {                          /*1方にしか動けない間*/
                mazeput( HEAD, body );                          /*  その方向に動き続ける*/
                v_add( HEAD, HEAD, &DIRECTION[s] );             /*  分かれ道にきたら*/
                mazeput( HEAD, body );                          /*  竜は死んでしまう*/
                v_add( HEAD, HEAD, &DIRECTION[s] );
        }
}


int     status( HEAD )                                          /*空いている方向を返す*/
vector  *HEAD;                                                  /*方向は、1方向なら*/
{                                                               /*  2次元なら1～4*/
        int     i, s=0;                                         /*  3次元なら1～6*/
        vector  NEWHEAD;                                        /*分かれ道では0を返す*/

        for ( i=1; i<=dimension*2; i++ ) {                      /*全部の方向を探る*/
                v_add( &NEWHEAD, HEAD, &DIRECTION[i] );
                if ( mazeget( &NEWHEAD )!=air ) continue;       /*行き先が空間なら*/
                if ( s==0 ) {                                   /*  それが戻り値だが*/
                        s=i; continue;                          /*  すでに空間をみつけてあれば*/
                } else {                                        /*  分かれ道と判断出来る*/
                        s=0; break;
                }
        }
        return( s );
}


void    printout()                                              /*ここはmaze2.cと同じ*/
{
        int     x, y, z;

        for ( z=0; z<size_z; z++ ) {
                for ( y=0; y<size_y; y++ ) {
                        for ( x=0; x<size_x; x++ ) {
                                switch ( maze[x][y][z] ) {
                                        case air:
                                                printf( " " ); break;
                                        case body:
                                                printf( "#" ); break;
                                        case wall:
                                                printf( "□" ); break;
                                }
                        }
                        printf( "¥n" );
                }
                printf( "¥n" );
        }
}


void    input( filename )                                       /*ファイルを読み込み*/
char    *filename;                                              /*  迷路を配列にしまう*/
{
        int             x, y, z, sx, sy, sz;
        FILE            *mazefile;
        unsigned char   line[maxsize_x+1];

        mazefile=fopen( filename, "r" );
        sz=0;
        for ( z=0; z<maxsize_z; z++ ) {
                sy=0;
                for ( y=0; y<maxsize_y; y++ ) {
                        fgets( line, maxsize_x+1, mazefile );   /*一行読み込み*/
                        sx=strlen( line )-1;                    /*  長さをとって*/
                        if ( sx>2 ) size_x=sx;                  /*  配列に入れる*/
                                else break;
                        for ( x=0; x<sx; x++ ) {
                                switch ( line[x] ) {
                                        case ' ':
                                                maze[x][y][z]=air; break;
```

▶マンレクと魔神宮を交互にやっていると「天国と地獄」が実感できる。とくに移植ものに多いが，操作性の悪すぎるゲームが横行している。老い先短いゲーマーにとって，トロいゲームは劇薬（触れるといっぺんで死んでしまう）である。　木村　信 (35) 神奈川県

```
					case '□':
						maze[x][y][z]=wall; break;
				}
			}
			if ( sx<3 && y==0 ) break;
			sy++;
		}
		if ( sy>2 ) size_y=sy;
		if ( feof( mazefile ) ) break;
		sz++;
	}
	size_z=sz;
	fclose( mazefile );
	if ( size_z==0 ) {					/*maze1.cで作った迷路だけを*/
		dimension=2;					/*　2次元として扱う*/
		size_z=3;					/*　処理しやすいようにずらす*/
		mazecopy(1,0);
	}
}


void	output( filename )				/*maze2.cとほぼ同じ*/
char	*filename;
{
	int	x, y, z;
	FILE	*mazefile;

	mazefile=fopen( filename, "w" );
	for ( z=0; z<size_z; z++ ) {
		for ( y=0; y<size_y; y++ ) {
			for ( x=0; x<size_x; x++ ) {
				switch ( maze[x][y][z] ) {
					case air:
						fprintf( mazefile, " " ); break;
					case body:
						fprintf( mazefile, "#" ); break;
					case wall:
						fprintf( mazefile, "□" ); break;
				}
			}
			fprintf( mazefile, "¥n" );
		}
		fprintf( mazefile, "¥n" );
	}
	fclose( mazefile );
}


void	mazecopy( z1, z2 )				/*maze1.cの迷路をずらす*/
int	z1, z2;
{
	int	x, y;

	for ( y=0; y<size_y; y++ ) {
		for ( x=0; x<size_x; x++ ) {
			maze[x][y][z1]=maze[x][y][z2];
		}
	}
}


void	v_add( V1, V2, V3 )				/*ここから先は*/
vector	*V1, *V2, *V3;					/*　maze2.cと同じ*/
{
	int	i;

	for ( i=0; i<3; i++ )
		V1->a[i]=V2->a[i]+V3->a[i];
}


void	v_mul( V1, V2, t )
vector	*V1, *V2;
int	t;
{
	int	i;

	for ( i=0; i<3; i++ )
		V1->a[i]=V2->a[i]*t;
}


unsigned char	mazeget( V )
vector	*V;
{
	int	x, y, z;

	x=V->a[0];
	y=V->a[1];
	z=V->a[2];

	return( maze[x][y][z] );
}


void	mazeput( V, c )
vector		*V;
unsigned char	c;
{
	int	x, y, z;

	x=V->a[0];
	y=V->a[1];
	z=V->a[2];
	maze[x][y][z]=c;
}
```

プチ・インタプリタを作ろう

Cによるサンプルプログラムとして，簡単なインタプリタ言語を作ってみるというのはいかがでしょうか。桒野氏のプログラムは関数ごとに豊富な注釈がつけられています。まずはこれを頼りに1週間ほどかけてプログラムをじっくりと読んでみてください。

Kuwano Masahiko **桒野　雅彦**

BASICの次はCの時代だ！　とばかりにあっちでもこっちでも大流行のC。私のCとのつきあいは，MS-DOS上で動くCコンパイラ（ちなみに名前をあげておくとOptimizing C86）が誰にも使ってもらえずラックの肥やしになっているのを見つけて以来ですから，3，4年といったところでしょうか。ホコリまみれの英文マニュアルと悪戦苦闘しながらROM化するユーティリティまでこしらえて機器組み込み用のプログラムを作ったものです。

そのCも，ちょっと前までは「あのUNIXのソースコードの大部分がCで書いてある」だったのが，聞き慣れたソフト名をあげて「○○はCで開発した」となってきたところを見ても，単なるお祭り騒ぎではなく，しっかりと根をおろしてきていることがうかがえます。特に4万円以下という低価格のCコンパイラが流通するようになってからは，ソフトハウスのようなプロの開発者だけでなく，アマチュアにも随分と普及してきているようです(かく言う私もXCを買ったひとりです)。

「Cはマシン語になるから速い」などと，理解に苦しむような会話ではしゃいでいるようなことはほとんどなくなり，生成するコードの質やライブラリの整備状況などに話がいきやすくなったところをみても，ホビーレベルにも，本気（マジ）に使われてきていることは確かなようです。

さて，われらがOh！Xでも改名以来初めて(!?)言語としてのCを大真面目に取りあげることになりましたが，形式ばかりの例題によるお勉強ではうんざりするというのは，周りに人がいて，前方でこちらに向かって話をしている人間がいると条件反射的に眠ってしまう私だけではないでしょう。なにより断片的な知識ではCのありがたみがわかりません。もっと実践のドキュメントをというOh！X編集室の意向もあり，まとまったプログラムをこしらえてみることになりました。

プチ・インタプリタ

あくまでもやや大きなサンプルといった性格のものですからあまり大袈裟なものを作ってもしかたがありません。とはいえ，ある程度それなりの行数は必要になるプログラムでないと面白くありません(Cの良さが生かされない)。そのあたりを踏まえてソースは500～600行，リスト用紙で10枚前後を目標にしてみました。もちろん作り方にもよりますが，このくらいあればある程度まとまった機能を果たすものが作れそうですし，10枚程度ならそれほど時間をかけなくても十分に全体を読み切ることができるでしょう。

さて，10枚程度で何を作りましょう。このくらいの枚数でも表計算，パーソナルデータベースなど，いろいろと作れそうですが，改造の楽しみやほかへの応用の広さということから，ちょっとしたインタプリタを作ってみることにしました。

言語仕様はBASICのサブセットですがあくまでも「勉強」が主体ということから，これ以上削ると，もはや「言語」と呼べるのか怪しくなるような「超サブセット」レベルのもので，とても「BASIC」とは呼べそうにありません。そこでBASICの名は使わず，「プチ(ちっぽけな)・インタプリタ」としておきましょう。最初は「Petty Interpreter」としようと考えていたのですが，この「Petty」はフランス語かなにかで「Petit(プチ)」となるという話を聞きつけ，耳ざわりのよさを買ってこうしました。頭文字を取って「PI(パイ)」。ちょっと可愛っぽく呼んでやってください。

このPIを1週間の作業で読んでみましょう。これが今回の「例題」です。

月曜日:言語仕様の決定

まず最初に言語としての仕様を決めてしまいましょう。先ほども述べたとおり，PIは勉強用ということで，命令体系はBASICの超サブセットとなっています。プログラム中に使える命令とそのフォーマットは次のとおりです。

```
LET      変数＝式
GOTO     式
GOSUB    式
RETURN
IF  式1  THEN  式2
PRINT    式
END
```

命令の意味についてはいまさら書くまでもないでしょう。もちろん実行はRUNです。さらに，インタプリタを簡単にするために，次のような制約を加えます。

- 扱える数値は10進数の整数だけです。
- 変数はアルファベットの並びで，頭の1文字だけで区別します。たとえばSKYPARKもSCARLETも同じ変数として扱われます。
- 式の中の演算子に優先順位はありません。したがって1＋1＊2の値は4になります。
- LET(代入命令)は省略できません。
- マルチステートメントは使えません。
- IF文は式1が0以外ならTHEN以下にある

図1　PIのサンプルプログラム

文字で波を描く

```
10 LET SOURCE=12345679
20 LET MULTI=9
30 LET FREQ=10
40 LET WAVE=SOURCE*MULTI
50 GOSUB 110
60 LET FREQ=FREQ-1
70 IF FREQ THEN 50
80 LET MULTI=MULTI+9
90 IF MULTI<82 THEN 30
100 END
110 LET CHILD=WAVE/100000000
120 LET LOOP=4
130 LET WAVE=WAVE/10
140 PRINT WAVE
150 LET LOOP=LOOP-1
160 IF LOOP THEN 130
170 LET LOOP=4
180 LET WAVE=WAVE*10
190 LET WAVE=WAVE+CHILD
200 PRINT WAVE
210 LET LOOP=LOOP-1
220 IF LOOP THEN 180
230 RETURN
```

パスカルの3角形

```
10 LET N=0
20 LET K=0
30 IF N=K THEN 100
40 LET R=K
50 GOSUB 1000
60 PRINT F;
70 LET K=K+1
80 GOTO 30
100 PRINT 1
110 LET N=N+1
120 IF N<11 THEN 20
130 END
1000 LET BUFF=N-R
1010 LET F=1
1020 IF N=R THEN 1100
1030 LET R=R+1
1040 LET F=F*R
1050 GOTO 1020
1100 IF BUFF=0 THEN 1200
1110 LET F=F/BUFF
1120 LET BUFF=BUFF-1
1130 GOTO 1100
1200 RETURN
```

式2の値の行に分岐するだけです。THEN以下に命令を書くことはできません。

- PRINT文は式の値を10進数で表示するだけの機能しかありません。
- 関数などは装備されていません。

特に4番目の「省略不可能なLET」というのはかなり凶悪な感じがするかもしれませんが，この制約は，1行のフォーマットを必ず，

行番号　命令　〈式など(命令はダメ)〉

となるようにして，少しでもインタプリタを作りやすくするためです。とはいいながら，IF文は少しこの形からはずれています。本当はTHENを省略したほうが簡単になるのですが，あまりにも体裁が悪くなるので，例外的に1行のなかで命令を2回チェックにいくようになっています。

火曜日:テキストフォーマットの決定

これから，いよいよインタプリタの制作にかかるのですが，なによりも先にプログラムを格納しておく形式を考えておかなくては，実行するにしても，どうしてよいのかわからなくなります。

PIでは，1行の構造は先頭の2バイトを行番号とし，それに続いてその行の文字列を格納する領域を固定長で42文字分とってあります。ここにヌルコード，すなわち“¥0”で終わる文字列を格納しておくわけです。この構造はプログラム中は，TEXTという構造体で定義しています。行番号を2バイトに圧縮したのは，エディタやGOTOなどの処理で行を見つけるのが楽になるようにするためです。

1行が固定長ですので，余ったところが無駄にはなりますがその代わり，Cの構造体が綺麗に使えるようになります。

あとは，プログラムの終わりを検出する手段が必要です。PIでは行番号で判別することにしました。入力できる行番号を0x7fff未満に制限し，0x7fff以上の数値が行番号として現れたらテキストの終了と見なすようにします。また，NEWコマンドを実行する関数clear_buffer(リスト1の114：〜)では，単純にすべての行の行番号をMAX_INTVAL(プログラム先頭の＃defineによりMAX_INTVAL→0x7fffと置き換え定義がなされている)にすることで，すべてを最終行の次の行番号すなわちテキストエンドにしてしまいます。この方法でNEWを実行するというわけです。

このプログラムテキストを表示するのがLISTコマンドの処理部であるlist_print out関数です(リスト1の220：〜)。for文一発で，簡単に済ませていますが，要は行番号がMAX_INTVAL未満だったら，その値を10進数で，さらにその行の文字列はそのままヌルコードがくるまで表示させるだけのことです。

水曜日:エディタの作成

昨日，テキストフォーマットを決めましたので今日はエディタを作ってしまうことにしましょう。2本あるリストのうちリスト1のPI MAIN.Cがエディタ部分です。エディタでは行の入力のほか，

LIST：テキストの表示
RUN：実行
NEW：テキストのクリア
EXIT：PIを終了する

の各コマンドを扱います。

main関数(リスト1の46：〜)に飛び込むと，初めにプログラムをクリアして“*Ready”のメッセージを出してから1行を入力するreadline関数(130：〜)を呼び出します。readlineでは1行を入力したら，小文字はすべて大文字に変換するなどの加工をしています。入力された行(テキスト)はline_buffer(42：で宣言されたchar型の配列)に格納されています。

次にget_command関数(80：〜)が呼び出されます。ここで入力行が先頭の文字からチェックされ，数字なら行番号が，LISTなどのコマンドなら，そのコマンド番号が帰ってきます。行番号はMAX_INTVAL(0x7fff)未満の正の数，コマンドの番号は負の数ですので，返ってきた値だけでなんであったかの区別がつけられます。

もし，数字だったならプログラムの入力ですから，put_text関数(148：〜)を呼び出して1行の入力を行います。

put_textでは，まずプログラムを格納しているtext_buffer(41：でTEXTという型の構造体として宣言されている)を見て，どこに挿入すればよいのかをチェックします。行番号は先頭行からだんだん大きくなっていきますから，text_bufferの各行の行番号を見て入力されたプログラムの行番号以上になったら，そこに入れることになるわけです。

行番号がそれまで入力されたどの行より大きい場合にはテキストの最後は行番号がMAX_INTVAL(0x7fff)で，行番号として許される値の最大値より大きいので，自動的にその場所，すなわち最終行に追加されることになります。

また，もし入力された行と同じ行番号があれば，その行を削除してから挿入を実行します。入力された行が行番号だけで，文字列がない場合には挿入は行いませんので，これだけで，挿入，書き換え，削除の3つのすべてに対応することになります。

木曜日:式の評価(その1)

言語仕様を眺めていて〈式〉の文字が多いのが目につきます。RETURN以外のすべての命令に式がからんでいます。案外気にして見ないと気づかないことなのですが，この類いの言語では式の扱いが大きなウエイトを占めているのです。まずは式の処理を攻略することにします。今日は括弧の処理はとりあえず後回しにして，通常の式を調べてみましょう。まず，典型的な式を書いてみましょう。

2＊3＋4－B

括弧の処理を除くと，式は一般的に〈数値(または変数)〉〈演算子〉〈数値(または変数)〉の並びを延々と繰り返していることがわかります。ここで，この処理をどう考えればよいか検討してみましょう。

PIで扱う演算子はすべて二項演算子，すなわち2つの値を入力として，ひとつの結果を出力するものです。しかも演算に優先順位がありませんから，処理は左から右への一方通行で進められます。〈値〉〈演算子〉〈値〉の形を処理し，計算結果をもって，今度は〈計算結果〉〈演算子〉〈値〉を計算し，その結果をもって……ということを延々と繰り返すことになります。

ただし式の頭だけは特殊で，ここでは演算子なしにいきなり出てきた数値を演算結果として扱い，その次の演算子の処理に入る必要があります。最初のうち，式の頭は必ず数値か変数であると仮定して処理を進めようとしていたのですが，式の頭にいきなり括弧が出てくる場合があるなど，案外いやらしいことがわかりました。この対策として今回は式の頭にはいつも“0＋”が隠れている(ダミーの“0＋”)として考えて処理を行うことにしました。そこまでの演算結果が0で演算子として“＋”があったようなふりをして演算を開始するのです。

たとえば，先ほどの式なら，

0＋2＊3＋4－B

となっていると考えるのです。先頭の値を取り込んで演算した結果が，取り込んだ値そのものになればよいので，1と掛け算を行っても構いませんが，この手のプログラムで掛け算を使うのはあまり好きでなかった

ので0との足し算にしました。

こうしておくと，足し算1回分だけ時間が無駄になりますが，処理としては統一された形になり，すっきりとした処理になります。

プログラムで，式の処理を行っているのはshiki(　)という関数(リスト2の251:)です。ここでは，そこまでの演算結果を変数accaに，演算子はop，次の値はaccbという変数に入力されることにしています。forループに入るところで，acca=0，op=OP_PLUSとやっているのが，先ほどの「ダミーの0+」です。

さて，ここで先ほどの式の例，2＊3+4－Bを処理させたとして考えていきましょう。いま，式を計算していって2＊3まで計算が終わったとします。この段階ではプログラムではswitch文(258：～)の直後にいることになります。ここでaccaには6が演算結果として入っています。

次に演算子を取り込むのですが，このとき取り込んだ文字が演算子でなければ演算を終了します。なお，演算子の取り込みはget_operater関数(325：～)で行っています。演算子を取り込んだら，今度はforループの先頭に戻ります。演算子があったなら，次は数値や変数がくるはずです。そこでget_value(286：～)を呼び出して値を取り込み，その値をaccbに代入します。

値と演算子が決まったので，続いてswitch文で演算子による分岐をして，演算を行います。

金曜日:式の評価(その2)

今日は括弧の処理を考えてみることにしましょう。括弧が入ると演算の流れが単純な左から右へという流れではなくなるため，少し気をつける必要があります。

括弧が付いた式というのは，

2＊(3+4)/(6－5)

のようなものです。人間がやるなら，まず括弧の中をすべて計算して，

2＊7/1

と変形してから計算するという技も使えますが，それをプログラムにするのは少々面倒です。

括弧付きの式を処理するもうひとつの方法は，まず(3+4)を計算して，2と掛けて，それから(6－5)を計算して，割り算をするというやり方です。この場合には“2＊”まで進んだところで，括弧があったら，一旦いま持っている“2”と“＊”は横に置いて，3+4の計算を進め，それが終わった時点で，そこで得られた値と先ほどの“2”と“＊”を使って計算することになります。

言葉で処理を言うと少々面倒なようですが，つまり昨日までの式の一般形の〈数値(または変数)〉のところに，「括弧でくくられた式」が追加されることになるのです。この，〈数値(または変数)〉を取り込むのは，get_value関数です。ですから，この処理に括弧を見つけたら式の計算を呼び出し，その計算値を値として返せばよいことになります。

ここで，get_value(286：～)を見てみましょう。3つ目のif文が括弧の処理です。括弧があると，現在の注目点を示すtextp(テキストポインタ)をひとつ進め，shiki関数を呼び出して括弧の中の値を計算します。計算が終わったら，閉じ括弧があるはずですので，とりあえずチェックしてリターンします。これでめでたく括弧の処理もできるようになりました。

ところで，get_valueはshikiから呼び出されていたはずです。そのget_valueが今度はshikiを使って値を得ているのです。つまり，get_valueというクッションが入ってはいますが，shikiがshikiを呼び出して処理をするわけです。どこかでこのような話を聞きませんでしたか？　そのとおり「再帰」です。数学の時間以外はパズルを解くときくらいしか縁のなかった再帰ですが，こうしてみると，ちょっとは使えそうな感じがしませんか？

土曜日:実行文の処理(その1)

いよいよ週末。テキストの入力も，式の処理も終わったのであとはいよいよインタプリタ本体です。本体はexecute関数です(リスト2の64：～)。最初のwhileで，最終行にくるまでループするようにしています。次に，行の先頭にある命令を取り込み(106：～のget_statement関数)，その番号に従って次のswitchで分岐します。

END

特に説明するまでもないでしょう。END AT…のメッセージを出して，ステータスをTERMINATEにすることによりwhileループを脱出し，実行を終了します。

LET

exec_let関数(129：～)で処理します。LETは，

〈変数〉=〈式〉

となっていますから，まず変数を識別します。次は必ず等号がきているはずです，そしてその次はお得意の〈式〉。式の計算をして，得た値を変数に代入しているのが最後のreturnの直前の，

variable[var] =val

です。

PRINT

exec_print関数(200：～)で処理しています。式の値を計算してprintfで表示し，もしも式の最後が“;”であったならそのまま，違うなら改行してリターンするだけです。

GOTO

exec_goto関数(155：～)で実行しています。まず行番号を見つけ，その位置をprogram.pointer(55:でTEXT型の構造体へのポインタとして宣言)にセットします。行番号はsearch_line関数(リスト1,163:～)によって探してきます。プログラムの実行はprogram_pointerに従い行っていますので，これを書き換えてしまえばGOTOになるわけです。

IF

予想されるとおり，exec_if関数(リスト2の166：～)で処理しています。IF文の書式は，

IF 〈式1〉 THEN 〈式2〉

ですから，まずshikiを呼び出して式1を計算します。その結果が真であれば(if(value))，次にTHENがあることを確認してから式2の処理ですが，これはGOTOとなんら変わるところがないのでGOTOの処理をそのまま拝借しています。

GOSUB

基本的にはexec_gosub関数(219：～)で処理することになっていますが，GOSUBはちょっと厄介です。というのはRETURNによって，GOSUBの直後に戻ってこなくてはならないからです。この処理のために，GOSUB専用スタック，exec_stackを設けました。GOSUB命令のあった行の次の行の位置をexec_stackに積み，RETURNではこのスタックを掘り返して，そこにジャンプするのです。帰り先をスタックに積むほかはGOTOと同じですから，ここでもexec_gotoを拝借しています。

RETURN

帰り先をスタックからひっぱり出してきて，program_pointerにセットしているだけです。

日曜日:あなたならどうする

Oh！Xでは安息日は定められていないので日曜日はデバッグ(！)となるのでしょうか。プログラムはなるべくややこしくならず，それでいてある程度はCらしい小技も

入っているようにということで，いろいろと悩みつつ，一気にデッチあげました。いきなりインタプリタですから，Cを覚えたての方には少々難しかったかもしれませんが，自分でぼちぼちとプログラムを作っていくようになれば，それほど苦労しなくても読み切れると思います。

入力に当たっては，XCやMS-DOS上で動くまっとうなCコンパイラなどを使う場合にはそのままでよいのですが，シャープのランゲージシリーズ（αC）を使う場合には少々手直しが必要です。

まず，コメントやメッセージの中にある漢字はコンパイラが誤解するので，取り払います。さらに，ランゲージシリーズでは構造体の配列が使えなかったので，ばらばらの配列に変更しています(struct CMD_TABLEのところ)。これに伴ってこの構造体へのアクセスを行っている部分を変更します。これらについては，リストの中でコメントで入れてあるので，よく見ながら変更してください。

さらに，ランゲージシリーズではどうしたわけか変数のextern宣言がうまく使えませんでした(私が悪いのかもしれないが)。これでは，エディタと実行部分でプログラムのバッファが共有できないので，ファイルの分割がうまくできません。しかたがないので，ファイル2本をまとめて1本にしてからコンパイルしてください。

＊　＊　＊

最後に，PI自身は本当に骨のような部分だけしかなく，命令の追加もやりやすくなっていますので，このまま拡張していって自分専用のインタプリタにするのもよいでしょう。

また，処理速度についても中間コードを使ったり，関数へのポインタを使って一気に分岐するなど，まだ改良の余地があります。慣れてきたら，そちらに手を加えたりするのも面白いと思います。

それではご健闘を。

リスト1　プチ・インタプリタPI(pimain.c)

```
/*-------------------------------------
  -  P I ・・・プチ・インタープリター  -
  -            コマンド処理部          -
  -------------------------------------*/
#define         EOS             '¥0'
#define         ERROR           -1
#define         READY           0
#define         MAX_INTVAL      0x7fff          /* 正の整数（行番号）の最大値   */
#define         CMD_NEW         -10             /* NEWコマンドのコード          */
#define         CMD_LIST        -11             /* LIST      〃                 */
#define         CMD_EXIT        -12             /* EXIT      〃                 */
#define         CMD_RUN         -13             /* RUN       〃                 */
#define         LINE_LENGTH     40              /* テキストの一行の長さ         */
#define         MAX_LINE        25              /* テキスト・バッファの行数     */

struct CMD_TABLE {                               /* コマンドテーブルの構造は     */
        char    *cmd_string;                     /* 文字列へのポインタと、       */
        int     cmd_number;                      /* コマンドのコードの羅列       */
        } command_table[] = {                    /* である                       */
                "NEW" ,CMD_NEW,
                "LIST",CMD_LIST,
                "EXIT",CMD_EXIT,
                "RUN" ,CMD_RUN,
                "",ERROR
        };
/*
 * ランゲージ・シリーズでは上のstructCMD_TABLEをこれと差し換えてね
 *
 * char *comd_string[5];
 * int  comd_number[5];
 */

struct  TEXT {                                          /* テキストの構造は             */
        short   line_number;                            /* 2バイトの行番号と            */
/*
 *  ランゲージ・シリーズ
 *      int     line_number;
 */
        char    line_text[LINE_LENGTH+2];               /*  それに続く文字列の並び      */
        };                                              /*（固定長としている）である    */
struct  TEXT    text_buffer[MAX_LINE+1];                /* 実際にプログラムが格納される         */
char    line_buffer[LINE_LENGTH];                       /* キーボードからの1行分のバッファ      */
char    *line_pointer;                                  /* 入力行の処理に利用する               */
int     text_lines;                                     /* 現在まで入力された行数       */

main()
{
        int     command_number;
/*
 * ランゲージ・シリーズ用の追加
 *      init_table();
 */
        clear_buffer();                                                 /* ゴミが入っていると好きくないのでクリア */
        for (;;) {
                printf("*Ready¥n");
                if (!readline(line_buffer,LINE_LENGTH)) {
                        printf("Input Error!¥n");
                        return(ERROR);
                }
                line_pointer = line_buffer;
                command_number = get_command();                         /* コマンド番号が0x7fff以下の正数なら行番号 */
                if ((command_number >=0 ) && (command_number < MAX_INTVAL))
                        put_text(command_number,line_pointer);
                else {
                        switch(command_number) {
                                case    CMD_NEW:        clear_buffer();
                                                        break;
```

```
                                case    CMD_LIST:       list_printout();
                                                        break;
                                case    CMD_EXIT:       printf("Bye...¥n");
                                                        return(READY);
                                case    CMD_RUN:        execute();
                                                        break;
                                default:                printf("??Command¥n");
                        }
                }
        }
}

/*              ****** コマンドのチェック *****
 *      line_pointerの指す先からの文字列のチェックを行う。
 *      数字ならば、行番号と見なし、違うならコマンドテーブル
 *      （command_table)から一致するものを捜しにいく。
 *      そんなコマンド無いわい！というときはエラーコードを返す。
 *              ほいほい。
 */
get_command()
{
        int     i,data;
        char    *c,*d;
        d = skipspace(line_pointer);
        if ((*d >= '0') && (*d <= '9')) {
                for (data=0; (*d >= '0') && (*d <= '9'); d++)
                        data = data * 10 + *d - '0';
                if ((data <= 0) || (data > MAX_INTVAL))
                        return(ERROR);
                line_pointer = d;
                return(data);
        }
        for (i=0; *(c = command_table[i].cmd_string) != EOS;i++) {
        /* ランゲージ・シリーズ用for (i=0; *(c = comd_string[i]) != EOS; i++) { */
                while((*c != EOS) && (*c == *d)) {
                        c++;
                        d++;
                }
                if (*c == EOS) {
                        line_pointer = d;
                        return(command_table[i].cmd_number); /* ランゲージ・シリーズ用return(comd_number[i]); */
                }
        }
        return(ERROR);
}

/*              *****  テキストバッファのクリアー *****
 *      と、聞くとたいそうな事をしているように思えますが、
 *      要は、行番号を全てMAX_INTVALにしているだけです。
 *      行番号の最大値はMAX_INTVAL-1に制限しているので
 *      それ以上の値が行番号として見つかったら、テキスト・
 *      エンドと見なすことができます。めでたしめでたし。
 */

clear_buffer()
{
        int     i;
        for (i=0; i<=MAX_LINE; i++)
                text_buffer[i].line_number = MAX_INTVAL;
        text_lines = 0;
}

/*              ***** 標準入力からの一行読み込み *****
 *      後の処理が簡単になるように、小文字は全て大文字に変換し、
 *      さらに、行の最後の改行コードは取り払い、EOSコードを入
 *      れるようにしています。
 */
readline(buffer, size)
        char    *buffer;
        int     size;
{
        int     status;
        char    *c;
        status = gets(buffer, size);
        for (; *buffer > 0xf; buffer++)
                *buffer = toupper(*buffer);
        *buffer = EOS;
        return(status);
}

/*              *****  テキストの挿入  *****
 *      入力しようとする行の行番号をテキストバッファから捜し、
 *      一致するものがあればそれをいったん削除してから挿入します。
 */
put_text(lnumber,input_text)
        int     lnumber;
        char    *input_text;
{
        int     line_counter;
        line_counter = search_line(lnumber);
        if (text_buffer[line_counter].line_number == lnumber)
                delete_line(line_counter);
        insert_line(line_counter,lnumber,input_text);
}

/*              ***** 行の検索 *****
 *      テキストバッファを最初から眺め、与えられた行番号（ｎ）以上の
```

```
 *      行を捜し、その行がテキストバッファの何行目であるかを返します。
 *      テキストの最後は行番号がMAX_INTVALになっているため、一番最後の
 *      行よりも大きい数がきてもうまくいくようになります。
 */
search_line(n)
        int     n;
{
        struct  TEXT    *text;
        int     i;
        for (i = 0,text = text_buffer; text->line_number < n; i++, text++ )
                ;
        return(i);
}

/*              ***** 行の削除 *****
 *      テキストバッファ上の行数(line_counter)をもらい、その行を削除します。
 *      要は、その行以降の行を一つづつ引き上げ、今まで最後だった行については、
 *      行番号をMAX_INTVALにするだけですが・・・。
 */
delete_line(line_counter)
        int     line_counter;
{
        for (; line_counter < MAX_LINE-1; line_counter++) {
                text_buffer[line_counter].line_number = text_buffer[line_counter+1].line_number;
                sprintf(text_buffer[line_counter].line_text,text_buffer[line_counter+1].line_text);
        }
        text_lines--;
        text_buffer[line_counter].line_number = MAX_INTVAL;
}

/*              *****  行の挿入  *****
 *      テキストバッファ上の行数(line_counter)をもらい、そこに一行
 *      （行番号はlnumber、中身はtext）を挿入します。行をずらして、
 *      sprintfではめ込むだけのことです。
 */

insert_line(line_counter,lnumber,text)
        int     line_counter,lnumber;
        char    *text;
{
        int     tail;
        if (text_lines < MAX_LINE) {
                if (strlen(text)) {
                        for (tail = MAX_LINE; tail > line_counter; tail--) {
                                text_buffer[tail].line_number = text_buffer[tail-1].line_number;
                                sprintf(text_buffer[tail].line_text, text_buffer[tail-1].line_text);
                        }
                        text_lines++;
                        text_buffer[line_counter].line_number = lnumber;
                        sprintf(text_buffer[line_counter].line_text, text);
                }
        }
        else .          printf("Sorry.. I can't eat so much text!¥n");
}

/*              ***** プログラムリストの表示 *****
 *      リストを表示します。例によって行番号がMAX_INTVALになったら、テキスト
 *      の最後と見なして終了します。
 */
list_printout()
{
        struct  TEXT    *p;
        char    *c;
        int     number;
        for (p = text_buffer; (number = p->line_number) < MAX_INTVAL; p++ )
                printf("%d%s¥n",number,p->line_text);
}

/*              ***** スペースのスキップ *****
 *      スペースやタブを読み飛ばします。
 */
skipspace(p)
        char    *p;
{
        while ((*p == ' ') || (*p == '¥t'))
                p++;
        return(p);
}

/*
 * ランゲージシリーズ用の追加
 *
 * init_table()
 * {
 *      comd_string[0] = "NEW";
 *      comd_string[1] = "LIST";
 *      comd_string[2] = "EXIT";
 *      comd_string[3] = "RUN";
 *      comd_string[4] = "¥0";
 *      comd_number[0] = CMD_NEW;
 *      comd_number[1] = CMD_LIST;
 *      comd_number[2] = CMD_EXIT;
 *      comd_number[3] = CMD_RUN;
 *      comd_number[4] = ERROR;
 *      init_exec();                    In  'PIEXEC.C'
 * }
 */
```


▶妻帯者の皆様はX68000の購入に苦労なさっているようですが，うちの嫁さんは私と一緒になってパソコンショップで粘ること4時間。サービスにサービスをつけさせて，ついでにボーナスの半分を支払ってくれることになりました。ばかぁー幸せもんだ……。

佐藤　啓市（33）北海道

リスト2　プチ・インタプリタPI(piexec.c)

```
/*--------------------------------------
  -  P I ・・・プチ・インタープリター   -
  -              実 行 処 理 部          -
  --------------------------------------*/
#define         EOS             '¥0'
#define         TAB             '¥t'
#define         FALSE           0
#define         TRUE            1
#define         READY           0
#define         ERROR           -1
#define         TERMINATE       1
#define         LINE_LENGTH     40
#define         MAX_LINE        25
#define         MAX_INTVAL      0x7fff
#define         OP_PLUS         1       /* '+' オペレータのコード      */
#define         OP_MINUS        2       /* '－'         〃             */
#define         OP_MUL          3       /* '＊'         〃             */
#define         OP_DIV          4       /* '／'         〃             */
#define         OP_EQUAL        5       /* '＝'         〃             */
#define         OP_HI           6       /* '＞'         〃             */
#define         OP_LO           7       /* '＜'         〃             */
#define         STATE_LET       1       /* ＬＥＴ命令のコード          */
#define         STATE_IF        2       /* ＩＦ         〃             */
#define         STATE_THEN      3       /* ＴＨＥＮ     〃（ＴＨＥＮが命令というのはちょいと苦しい気がする）    */
#define         STATE_GOTO      4       /* ＧＯＴＯ     〃             */
#define         STATE_GOSUB     5       /* ＧＯＳＵＢ   〃             */
#define         STATE_RETURN    6       /* ＲＥＴＵＲＮ 〃             */
#define         STATE_PRINT     7       /* ＰＲＩＮＴ   〃             */
#define         STATE_END       8       /* ＥＮＤ       〃             */
#define         STACK_SIZE      10      /* ＧＯＳＵＢ用スタックのサイズ */

                                        /* テキストと命令テーブルの構造 */
struct  TEXT {                          /* CMD.Cのところで頑張って書いたからそちらを見てね     */
        short   line_number;
        char    line_text[LINE_LENGTH+2];
        };

struct          CMD_TABLE {             /* ランゲージ・シリーズ         */
        char    *cmd_string;            /* char *state_string[9];       */
        int     cmd_number;             /* int  state_number[9];        */
        } statements[] = {              /*                              */
                "LET",STATE_LET,        /*                              */
                "IF",STATE_IF,          /*                              */
                "THEN",STATE_THEN,      /*                              */
                "GOTO",STATE_GOTO,      /*                              */
                "GOSUB",STATE_GOSUB,    /*                              */
                "RETURN",STATE_RETURN,  /*                              */
                "PRINT",STATE_PRINT,    /*                              */
                "END",STATE_END,        /*                              */
                "",ERROR                /*                              */
        };

extern  struct  TEXT    text_buffer[];  /* ランゲージ・シリーズの時はこれを削除します   */

struct  TEXT    *program_pointer;       /* 現在、どの行を実行しているのかを指し示すポインタ     */

int     exec_stack[STACK_SIZE];         /* GOSUB RETURNの処理のためのスタック                   */
int     stack_pointer;                  /*            〃            スタックポインタ            */
int     terminate;                      /* 式の処理でエラーが起こったときに立てるフラグ         */
int     variable[26];                   /* 変数領域。Ａはvariable[0],Ｂはvariable[1]・・・      */
char    *textp;                         /* 現在処理している文字の位置を示すポインタ             */


/********************** 実行ルーチン：はじまりはじまり ********************/
execute()
{
        int     status,i;
        terminate = FALSE;                                                      /*  あれやこれやの初期化 */
        stack_pointer = STACK_SIZE;
        program_pointer = text_buffer;
        status = READY;
        while ((program_pointer->line_number < MAX_INTVAL) && (status == READY)) { /*  テキストの終わり(MAX_INTVAL以上の
                                                                                       行番号)に来たらおしまい）*/
                i = get_statement(program_pointer->line_text);
                switch (i) {
                        case    STATE_LET:      if ((status = exec_let()) == READY)
                                                        program_pointer++;
                                                break;
                        case    STATE_IF:       status = exec_if();
                                                break;
                        case    STATE_GOTO:     status = exec_goto();
                                                break;
                        case    STATE_GOSUB:    status = exec_gosub();
                                                break;
                        case    STATE_RETURN:   status = exec_return();
                                                break;
                        case    STATE_PRINT:    if ((status = exec_print()) == READY)
                                                        program_pointer++;
                                                break;
                        case    STATE_END:      printf("*END at [%d]¥n",program_pointer->line_number);
                                                status = TERMINATE;
                                                break;
                        default:                printf("??Statements¥n");
                                                status = ERROR;
                }
        }
        if (status == ERROR)                                                 /*  エラーのあった行を表示して終了  */
```

▶渡辺美里がアイドルでないと思っている読者諸氏へ。いまはロックで売っているけど，前はミスセブンティーンからスクールメイツに入っていたことを隠しているのは，どうも私にはいただけません。　佐藤　正貴（18）千葉県

```
			printf("[%d%s]¥n",program_pointer->line_number,program_pointer->line_text);
}

/*			***** 命令の判別 *****
 *	与えられたポインタ(string)の先にある文字列と命令テーブル(statements)
 *	のコマンド表を比べて、一致するものを捜す。
 *	無かった場合にはエラーコードを返す。
 */
get_statement(string)
	char	*string;
{
	char	*c,*d;
	int	i;
	string = skipspace(string);
	for (i=0; *(c = statements[i].cmd_string) != EOS; i++) {
	/* ランゲージ・シリーズでは for (i=0; *(c = comd_string[i]) != EOS; i++) { */
		for (d=string;(*c != EOS) && (*c == *d);c++,d++)
			;
		if (*c == EOS) {
			textp = skipspace(d);
			return(statements[i].cmd_number); /* ランゲージ・シリーズではreturn(state_number[i]); */

		}
	}
	return(ERROR);
}

/*			***** 代入文の処理 *****
 *	代入文は　変数　=　式　の形である。
 *	（こんな風に書くと偉そうでしょう？）
 */
exec_let()
{
	int	var,val;
	textp = skipspace(textp);				/* 変数 */
	if ((*textp < 'A') || (*textp > 'Z')) {
		printf("変数が必要です¥n");
		return(ERROR);
	}
	var   = *textp-'A';
	textp = skipalpha(textp);
	textp = skipspace(textp);
	if (*textp++ != '=') {					/* =	*/
		printf("等号が必要です¥n");
		return(ERROR);
	}
	textp = skipspace(textp);
	val = shiki();						/* 式	*/
	if (terminate)
		return(ERROR);
	variable[var] = val;
	return(READY);
}

/*			***** ＧＯＴＯの処理 *****
 *	ＧＯＴＯ　式　の形で、式の値が行番号になる。
 */
exec_goto()
{
	int	n;
	textp = skipspace(textp);
	n = search_line(shiki());				/* 式の値を行番号として該当する行を捜してくる	*/
	if (terminate)						/* 式に誤りがあれば勿論、エラー			*/
		return(ERROR);
	program_pointer = text_buffer+n;			/* プログラム・ポインタをセットすればジャンプしてくれる */
	return(READY);
}

/*			***** ＩＦ・・ＴＨＥＮの処理 *****
 *	ＩＦ　式１　ＴＨＥＮ　式２
 *		式１の値が０以外なら式２の行へ飛ぶ。
 *	行番号しか記述できないところが、昔のＢＡＳＩＣっぽい、レトロ感覚である。
 */
exec_if()
{
	int	value;
	textp = skipspace(textp);
	value = shiki();						/*  式１		*/
	if (terminate)
		return(ERROR);
	if (value) {
		textp = skipspace(textp);
		if (get_statement(textp) != STATE_THEN) {		/*  ＴＨＥＮ	*/
			printf("THENがありません¥n");
			return(ERROR);
		}
		return(exec_goto());					/*  式２に飛ぶ	*/
	}
	else	program_pointer++;
	return(READY);
}

/*			***** ＰＲＩＮＴ *****
 *	ＰＲＩＮＴ　式			式の値を表示して改行する
 *	ＰＲＩＮＴ　式；			式の値を表示したあと、改行しない
 *
 *  .......... おまけ ........
 *	殺しているＷＨＩＬＥ文を復活させると、次のような書き方もできるようになる。
 *
 *	ＰＲＩＮＴ　式１　式２　式３・・		式１の値を表示したあと、改行して式２、式３の値を表示する
 *	ＰＲＩＮＴ　式１、式２、式３　・・		〃			改行せずに式２、式３の値を表示する
```

▶田村さん，実は僕もモデル10ユーザーだったのです。苦しい生活のなかでやっとディスクを買うだけのお金がたまったときはたと気がついたのです。……こ，これは，X1turbo IIが買えるのではないか，と。現在，僕はX1turboZのユーザーです。なにかうしろめたいものを感じながらも，楽しい生活を送っています。　八尋　秀治（17）福岡県

```
 */
exec_print()
{
        textp = skipspace(textp);
        printf("%5d",shiki());
/*
 *      while(!terminate && ((*textp == ',') || (*textp == ' ')) ) {
 *              if (*textp == ' ')
 *                      printf("¥n");
 *              textp = skipspace(++textp);
 *              printf("%5d",shiki());
 *      }
 */
        if (terminate)
                return(ERROR);
        if (*textp != ';')
                printf("¥n");
        return(READY);
}

/*              ***** ＧＯＳＵＢの処理 *****
 *      ＧＯＳＵＢ　式
 *              式の行を呼び出します。
 *      帰り先（要するに次の行）をスタック(exec_stack)に積み上げる他は
 *      ＧＯＴＯと同じである。
 */
exec_gosub()
{
        if (stack_pointer <= 0) {                               /* スタックのチェックをして・・・        */
                printf("ネスティングが深すぎます¥n");
                return(ERROR);
        }
        exec_stack[--stack_pointer] = ++program_pointer;        /* 帰り先（次の行）をスタックに積んで・ */
        return(exec_goto());                                    /* ＧＯＴＯをそのまま拝借！簡単！簡単！ */
}

/*              ***** ＲＥＴＵＲＮの処理 *****
 *      ＲＥＴＵＲＮ
 *              ＧＯＳＵＢで呼ばれたところに戻る
 *      帰り先はスタックの中にある。これをひっぱり出して、program_pointer
 *      に代入すればそこにジャンプしてくれます。ホイサッサ
 */
exec_return()
{
        if (stack_pointer >= STACK_SIZE) {                      /* 念のためにチェック                   */
                printf("戻り先がありません¥n");
                return(ERROR);
        }
        program_pointer = exec_stack[stack_pointer++];  /* スタックからひっぱり出してprogram_pointerにセット    */
        return(READY);                                  /* めでたしめでたし                                     */
}

/******************** 式の計算だよ ********************/

shiki()
{
        int     acca,accb,op,stat;
        for (acca = 0,op = OP_PLUS,stat = READY; (stat == READY) && !terminate; ) {
                accb = get_value();
                switch(op) {
                        case    OP_PLUS:        acca += accb;
                                                break;
                        case    OP_MINUS:       acca -= accb;
                                                break;
                        case    OP_MUL:         acca *= accb;
                                                break;
                        case    OP_DIV:         acca /= accb;
                                                break;
                        case    OP_EQUAL:       acca = (acca == accb);
                                                break;
                        case    OP_HI:          acca = (acca > accb);
                                                break;
                        case    OP_LO:          acca = (acca < accb);
                                                break;
                        default:                terminate = TRUE;
                                                printf("Operater error¥n");
                }
                op = get_operater(&stat);
        }
        if (stat != TERMINATE) {
                terminate = TRUE;
                printf("??ERROR¥n");
                return(-1);
        }
        return(acca);
}

/*              *****  項の値を取ってくる *****
 *      変数や数値が主だが、括弧があったときには、ちょっと注意。
 *      括弧が閉じるまでが一つの項であるからshikiを呼び出して
 *      括弧の中の値を計算する。
 */
get_value()
{
        int     value;
        char    c;
        value = 0;
        c     = *textp;
        if ((c >= '0') && (c <= '9')) {                         /* 数値 */
                for (;;c = *++textp) {
                        if ((c >= '0') && (c <= '9'))
```

▶6月号の質問箱にあった中谷さんにひと言。BASCOM で漢字を使うにはコンソール関係をすべて自作のマシン語ルーチンにしてCALL命令で呼び出せばよいと思います。実際，私はそうやっていますから。私の持っているBASCOMではライブラリにあるコンソールモジュールをかなり書き換えないとマスクがはずれないようです。　沼野　博 (23) 東京都

```
                                value = value*10+c-'0';
                        else    break;
                }
                return(value);
        }
        if ((c >= 'A') && (c <= 'Z')) {                         /* 変数 */
                textp = skipalpha(textp);
                return(variable[c - 'A']);
        }
        if (c == '(') {                                         /* 括弧 */
                textp++;
                value = shiki();                                /*  式を計算して・・    */
                if (*textp != ')') {                            /*  勿論、括弧は閉じていなければいけない        */
                        terminate = TRUE;
                        printf("括弧の数が合わないと思うのですが・・・¥n");
                        return(ERROR);
                }
                textp++;
                return(value);
        }
        terminate = TRUE;                                       /* それ以外ならエラー   */
        printf("数値の筈なんですが・・・¥n");
        return(ERROR);
}

/*              ***** 演算子の取り込み *****
 *      ';'で終了させるのは、ＰＲＩＮＴ命令で使うため（式の最後が；なら改行しない）
 *      ','でも 〃                        で数値の連続表示ができるパッチに対応させておくため
 */
get_operater(status)
        int     *status;
{
        int     operater;
        *status = READY;
        switch(*textp) {
                case    '+':    operater = OP_PLUS;
                                break;
                case    '-':    operater = OP_MINUS;
                                break;
                case    '*':    operater = OP_MUL;
                                break;
                case    '/':    operater = OP_DIV;
                                break;
                case    '=':    operater = OP_EQUAL;
                                break;
                case    '>':    operater = OP_HI;
                                break;
                case    '<':    operater = OP_LO;
                                break;
                case    ';':
                case    ',':
                case    ' ':
                case    TAB:
                case    EOS:    *status  = TERMINATE;
                                break;
                default:        printf("演算子がおかしいように思うのですが・・・¥n");
                                *status = ERROR;
                                return(ERROR);
        }
        if (*status == READY)
                textp++;
        return(operater);
}

/*              ***** アルファベット文字（大文字）のスキップ *****
 *      変数名に冗長を許したりするために作っておいた
 */
skipalpha(p)
        char    *p;
{
        while ((*p >= 'A') && (*p <= 'Z'))
                p++;
        return(p);
}

/*
 * ランゲージ・シリーズ用の追加
 *
 *init_exec()
 *{
 *      state_string[0] = "LET";
 *      state_string[1] = "IF";
 *      state_string[2] = "THEN";
 *      state_string[3] = "GOTO";
 *      state_string[4] = "GOSUB";
 *      state_string[5] = "RETURN";
 *      state_string[6] = "PRINT";
 *      state_string[7] = "END";
 *      state_string[8] = "¥ 0";
 *      state_number[0] = STATE_LET;
 *      state_number[1] = STATE_IF;
 *      state_number[2] = STATE_THEN;
 *      state_number[3] = STATE_GOTO;
 *      state_number[4] = STATE_GOSUB;
 *      state_number[5] = STATE_RETURN;
 *      state_number[6] = STATE_PRINT;
 *      state_number[7] = STATE_END;
 *      state_number[8] = ERROR;
 * }
 */
```

特別講義1 XBAS to Cの正しい使い方

Murata Toshiyuki
村田 敏幸

XCにはXBAS to Cという秘密兵器が隠されています。X-BASICのプログラムをCのソースに変換してコンパイルできるわけですが，正しく使ってBASICコンパイラとして活用するには，まずCとBASICとの性格の違いを理解することが大切のようです。

みんなも知っているようにX-BASICはC言語に近い制御構造を持つという意味でCとはかなり微妙な関係にある。この微妙な関係に拍車をかけるのがC compiler PRO-68K(通称XC)を買うとくっついてくるXBAS to Cコンバータ(BC)の存在だ。

BASICで書いたプログラムもBCを通せばCのプログラムに早変わりだ。Cに変換してしまえばこっちのもんで，最初からCで書いたプログラムと同じようにコンパイルすれば実行ファイル（.x）ができあがる。.x＝マシン語のプログラムということだから，BASICインタプリタがなくても実行できるし，なにより速い。BASICインタプリタ上では遅くて使いものにならなかったようなプログラムもコンパイルすればビュンビュン走る。今まではまともなプログラムを開発するにはマシン語なりCなりを覚えなければならなかったけれど，BC(を含むXC。以下同様)を使えば僕らのいちばん身近にある言語，BASICで「開発」が行えてしまうのだ。

手順も至って簡単で，たとえばtest.basというプログラムからtest.xを作るにはコマンドモードで，

```
cc test.bas
```

と打ち込むだけでいい。

このCCというのはコンパイルドライバと呼ばれるプログラムで，CCのあとに続けてBASICのファイル名を書けばCへの変換，コンパイル，アセンブル，リンクという手順を自動的に行ってくれる。念のためだけど，CC自体がコンパイルしたりアセンブルしたりといった機能を持っているわけじゃなく，CCはCコンパイラやアセンブラなんかを呼び出すだけの仕事しかしていない。だから「ドライバ」というんだけど，これは覚えてくれなくてもいい。とにかく，XCはあまりCを知らない人にもBASICコンパイラ感覚で使うことができる便利なツールだってことはわかってもらえたと思う。

ここまでは，なんだかとってもおいしそうな話で，「Cはちょっと……」と尻ごみしていた人もXCがほしくなったんじゃないだろうか。実際，XCを買った人の何割かはBASICプログラムをCに変換する機能が目当てなんだろうね。けれども，うまい話には落とし穴があるのは世の常だ。BASICコンパイラを買ってきたつもりでXCを使うと確実にハマる。

やってみた人にはわかると思うけれど，「インタプリタで動作が確認されていて，Cにもすんなり変換できて，コンパイル，アセンブル，リンクもエラーなしで通ったにもかかわらず，最終的に得られた.xファイルは正しく動作しない」ということもある。こうなってしまうと，Cについて詳しくない人にはもうどうすることもできないわけだよね。

やっぱりCを覚えなければいけないんだろうか，と，不安になった人もいると思う。なにも考えないよりはちょっとぐらい謙虚なほうが僕は好きだけど，はっきりいってBCを使うだけならCを覚える必要は全然ない。いくつかのことを「体で覚えて」それを守れば，Cを知らなくてもBCを使うことはできるんだ。この際，理屈はあと回しでもいいことにしよう。気になる人は自習してちょうだい。

マクラが長くなったけど，そんなわけで以下，X-BASIC+BCでプログラムを作るうえでの注意点をなかば対症療法的に書いてみる。

あくまでもCコンパイラ

最初に少々おせっかいだけれど，心構え(重い言葉だね)の話から始めよう。上でもチラチラ書いたように「BCはBASICコンパイラのようにも使えるけど，決してBASICコンパイラではないんだ」ということを頭に入れておいてほしい。一度Cに変換されてからコンパイルされるのだということを忘れてしまうと思わぬバグを生む結果となるからね。

そもそも，X-BASICがどんなにCに似ているように見えても，結局はBASICなんだ。さらに，BASICとCの違いに加えて，インタプリタとコンパイラという実行方法の違いが加わるから話は複雑になる。

BASICインタプリタはかなりルーズな言語処理系だ。よくいわれるのはデータの型の区別が（あまり）ないという点で，X-BASICの場合は結構データ型にはうるさいほうなんだが，それでもルーズなことには変わりない。たとえば，実数の引数を取る関数に整数を渡すことができてしまう。この場合インタプリタは「あれ？　引数は実数のはずなんだけどなぁ。しかたがない。私が骨を折りましょ」と，実数に変換してから関数に渡すので，ちゃんと望むとおりの結果が返ってくる。ユーザーはちょっとぐらいルーズでも構わない。

対してCコンパイラはまた別の意味でルーズだ。やはり，実数を引数に取る関数に整数を渡すことができる。が，型の変換なんかはしないで，そのまんま関数に渡してしまう。引数を受け取る関数側では渡されたのは実数だと思って処理をするから，どんな結果が出るかは見当もつかない。コンパイラがルーズなわけだ。

たったこれだけを見ても，BASICインタプリタとCコンパイラとの間には暗くて深い河があるのがわかるというもの。そのへんを踏まえて以下を読んでほしい。

見た目はすんなり通るけど……

さて，もっとも多くの人が落っこちてもがいていると思われる落とし穴が，何を隠そう，前項で例にあげた関数の引数の問題だ。この問題のやっかいなところは，コンパイル（Cへのコンバートを含む。以下同様）は見たところ正常に行われるという点にある。

例をあげよう。X-BASIC上で2つの数の大きいほうを返す関数maxを作ったとする。

〈例1〉

```
1000 func float max(a;float, b;float)
1010      if (a>=b) then return(a)
1020      return(b)
1030 endfunc
```

引数も返り値も実数型になっているのがポイントで，この関数をインタプリタ上で使う分には，引数に整数型の値を与えても，もちろん実数型の値を与えても正しい答えが返ってくる。たとえば，

```
100  int i=5,j=6,k
```

110　k＝max(i, j)

とすれば，kにはちゃんと6が代入される。

ところが，これをコンパイルしてみると，突拍子もない答えが返ってきて驚くことになる。すでに述べたようにX-BASICインタプリタでは関数側で宣言された仮引数の型に実引数の型を合わせてくれるが，Cでは関数呼び出しの際に型の変換を行わないからこういうことが起こるわけだ。

この例では仮引数を実数型で宣言しておきながら，整数型の変数を渡すという単純なミスだった。ちょっと注意すれば未然に防げるバグといえる。では，これはどうだろう。

100 float f, g

110 g＝max(f, 3)

一見正しいプログラムに見える。が，しかし，このプログラムをコンパイルしてみるとやっぱり変な答えが返ってくる。なぜかというと，ポコンと書いた3という数が整数とみなされてしまうからなんだ。

思い出してほしい。X-BASICでは数値は特に指定されていなければ整数として扱われるのだった。これはCでも同じだ。インタプリタ上では例によって型変換をしてくれるからこのことは表には出てこない。でもBCでCへ変換するとやっぱり整数なわけだから，ちゅどーんしてしまう。

この場合，

110 g ＝max(f, 3＃)

と書いておけば，BCにも実数だということが伝わるから，実数として扱われるようにコンバートしてくれる。

この応用というか，裏技みたいなものだが，実数を受け取る関数に整数型の変数値を渡すときには，

キャスト演算

異なる型の変数を混在して扱いたい場合，Cではキャストという手法を使うことで，型を強制的に変更することができる。〈例1〉の場合だと，

k＝max((double)i, (double)j);

とすれば，関数には実数に変換された値が渡される。また，最新のANSI仕様では関数のプロトタイプ宣言というのが導入されていて，double max (double, double)；という宣言をあらかじめ(この関数を使う前に)書いておけば，maxの引数がdouble型(倍精度実数型＝X-BASICのfloat型)かどうかをコンパイラがチェックしてくれるようになる。

XCは「ANSI仕様も取り入れた最新バージョン(シャープの広告によると)」で，プロトタイプ宣言もサポートされている。にもかかわらず，BCはプロトタイプ宣言はおろか，キャストさえしてくれないんだから悲しい。なんとかバージョンアップしてもらいたいものだ。

110 k＝max(i＋0＃, j＋0＃)

のように書いておけば，コンパイルの際にi, jを実数に変換してくれるようになる。これは型の違う値同士の演算のときには高いほうの型に変換されるというCの規則を利用したものだ。もっとも，みっともないからあんまり多用は勧めないけどね。逆に整数を受け取る関数に実数を渡すときには素直にint関数を使えばよい。

なお，ここに書いたことはユーザー定義関数だけではなく，システムがもともと持っている関数にも当てはまる。特に危ないと思われるのが，sgnとabsで，どちらの関数とも引数，返り値ともに実数型だ。僕も前に整数の符号を調べようとして引っ掛かったことがある。

ところで，「型の違う変数間の代入はどうなんだ」と不安に思った心配性の人もいるかもしれないので補足しておくけど，いかにCが不親切とはいえ，代入のときは自動的に型変換を行ってくれる。

だから，

100 int i＝5

110 float f

120 f＝i＋0＃

なんてことはしないで，ただ，

120 f＝i

と書けば，コンパイルしても f には実数の5が代入される。念のため。

文字列を扱うときのお話

CはBASICと比べるとデータ型の種類がずっと多い。ところが，面白いことにBASICにはあるのが当たり前なのにCにはないデータ型もある。文字列型がそうだ。Cには文字変数というものが存在しない。ただ文字列が扱えないかというと，そうじゃなくて，文字列はchar型の配列という形で表すことになっている。配列だから代入をしたければ配列の各要素をそれぞれ代入しなければならない。これではあまりに面倒なので，実際には文字列をコピーする関数が用意されている。BCはこのへんのことを踏まえて，文字列の代入が出てきたら文字列をコピーする関数を呼び出すようなCプログラムに変換するわけだ。文字列の連結や部分抽出も同じように処理される。

というわけで，文字列の扱いについてはCへコンバートするからといって特に気をつかう必要はない。もちろん，これはインタプリタ上で動作を確認してあればの話。ろくにデバッグもしていないと，なにかの勘違いで，

100 int i

110 str s

⋮

1000 i＝s

なんて部分があったりするかもしれない。これをいきなりCにコンバートしたとすると，char型の配列を整数型の変数に代入するというわけのわからないプログラムになってしまう。Cではこんなことをしてもよくって（少なくともしちゃいけないことはない)，今の例ではchar型の配列sの先頭アドレスがｉに代入される。チョンボと紙一重なのだが，こういうことをすることもあるのだ。くれぐれも，インタプリタでデバッグしてないプログラムをコンパイルするのはやめようね。

あ，忘れるところだった。X-BASICでは許されてCでは使えない文字列がらみの話があったんだっけ。でも，これはよい子は必ず読んでいる「Cユーザーズマニュアル」に書いてあることだから，説明しなくてもいいかな？　やっぱりしておこう。

まず，次の例のようにswitchやcaseの後ろに文字変数や文字列を書いてはいけない。

〈例2〉

```
100 str s＝"ABC"
110 switch s
120    case "ABC"：～
130    case "abc"：～
140          default：～
150 endswitch
```

この書式はインタプリタでは許され，結構便利なんだけど，さっきもいったようにCに変換されたときには s はchar型の配列になるんだから，意味不明のプログラムとなってしまう。

100 dim char s(32)

110 switch s

なんて書く人はいないでしょ？

もうひとつ，やっちゃいけない文字列操作としてマニュアルに書いてあるのが，

strlwr

strupr

strrev

の引数に文字列を直接記述してはいけないということ。つまり，

1000 strrev("ABCDEFG")

というような書き方はペケだというわけなんだけど，X-BASICインタプリタではそもそもこれら3関数の引数は「文字変数」でなければならないことになっているから問題ないだろう。ただ，このような書き方がCでは許されてしまうから気をつけなさい，という程度に解釈しておけばいい。

これが式にまつわる落とし穴だ

次に式まわりに触れておく。X-BASICとCで致命的に違うのが論理式の値だ。X-BASICでは真のとき−1，偽のとき0を値とするが，Cでは偽は同じく0なんだけど，真のときは1になってしまうんだ。試しに，

10 print (1=1)

という1行プログラムをコンパイルして，X-BASIC上で動かしたときと比べてみてほしい。

この違いはどんなときに問題になるかというと，BASIC慣れした人が好んで使う次のようなパターンがその代表だろう。

1000 a=a+(a>10)＊10

このプログラムはaが10より大きければaから10を引くということで，

1000 if a>10 then a=a−10

と同じ意味になる。aが10より大きければ，

(a>10)

の値が−1になり，それを10倍したものをaに足すのだからaは10小さくなる。ところがこれをコンパイルすると，aが10より大きければ，

(a>10)

は1となり，10引くつもりが10を足すことになってしまうのだ。

別のパターンとしては，こんなのもある。

〈例3〉

100 int a, c, fp

110 fp=fopen ("test", "r")

120 c=fgetc (fp)

130 a=islower (c)

140 if a=−1 then～

このプログラムはtestというファイルから1文字読み込んで，それが英小文字だったら140行のthen以下の処理をするというものだが，コンパイルすると思ったような動作をしない。読み込んだ文字が小文字のときにislowerの返す値が1になってしまうため，140行の条件式が常に偽となるからthen以下が決して実行されなくなってしまうのだ。

このようなことを避けるためには140行を

140 if a＜＞0 then～

としなければいけない。「もし真だったら」という条件を「もし偽じゃなかったら」に変えるわけ。意味は同じだよね。

ここでちょっと余計な話をしておこう。X-BASICの関数のなかにはエラーのときに−1を返すものがある。これはあくまで「エラーのときは−1」であって「真を表す−1」とは全然違う話だ。だから，コンパイルしても−1が1になるようなことはないので注意してほしい。

さて，式に関してX-BASICとCで違う点の2番目は演算子の優先順位だ。といってもさすがにCでも＊より＋のほうが優先されるなんてことはない。マズいのはシフト演算，比較演算なんかを使ったときで，これらの場合，演算の順序が違ってしまうことがある(とマニュアルに書いてある)。明らかに違うのはnotの優先順位で，X-BASICでは，

1000 if not a=2 then ～

という1文は，

1000 if not(a=2) then ～

の意味だけど，Cへコンバートされるときには，

1000 if (not a)=2 then～

と見なされてしまう。BCを通すときには過剰なぐらいカッコをつけたほうが安心ということだ。

マニュアル，ちゃんと読んでる?

「Cユーザーズマニュアル」の92ページに「名前のつけかた」という項がある。ここではC言語に変換する場合，プログラム中で使ってはいけない名前として，

- main
- Sで始まり6桁までの半角数字が続いている名前
- Lで始まり6桁までの半角数字が続いている名前
- b_で始まる名前

が，また，注意の必要な名前として，

- $で終わっている名前

があげられている。マニュアルから落ちているようなので，もうひとつ使わないほうがいい名前として，

- strtmp0……strtmp9

をこの場でつけ加えておく。

使っちゃいけないと書いてあるんだから使わなければいいわけなんで，これは大きな問題にはならないだろう。もし間違って使ってもコンパイル時かアセンブル時にエラーが出るから，それから直せばすむ。話はこれで終わりなんだけど，ついでだから好奇心旺盛な人のために，どうして使っちゃいけないかという話をしておくとしよう。

まず，mainという名前はCでは特別な関数名として扱われていて，プログラムのどこかに必ずmainという関数がなければならず，どこで宣言されていようとプログラムの実行はmainから始まることになっている。X-BASICの場合，メインルーチンは関数の形をしていないけど，Cへコンバートされるときにmainがどこにもないと困るので，メインルーチンをmainという名前の関数にしてしまうわけだ。ユーザーがmainという名前を別に使ってしまうと，ダブってしまうわけだから当然エラーになる。例によって念のため補足しておくと，mainという名前の「局所変数」は使っても構わない。わかるよね。

大文字のSとLから始まって残りは数字だけでできている名前はgosubやgoto

図1　BCの使用例

リスト1：X-BASIC上では正しく動くのにコンパイルすると動かないプログラム

```
 10 /* コンパイルすると動かない例 */
 20 str s
 30 test(" π  =",1,1)
 40 test(" π/2 =",2,0.5#)
 50 test(" 2π =",3,2)
 60 print "----------hit key----------";
 70 s=inkey$
 80 print
 90 end
100 /*
110 func test(mes;str,no,mul;float)
120     print no;")";
130     print mes;
140     print pi(mul)
150 endfunc
```

リスト2：リスト1をBCにかけたもの

```
#include        "basic0.h"
static  unsigned char   s[32+1];
int     test();

/********** program start **********/
void
main(b_argc,b_argv)
int     b_argc;
char    *b_argv[];
{
unsigned char strtmp0[256];
b_init();
 /* コンパイルすると動かない例 ***/
 test(" π  =",1,1);
 test(" π/2 =",2,0.5);
 test(" 2π =",3,2);
 b_sprint("----------hit key----------");
 b_strncpy(sizeof(s),s,b_inkeyS(strtmp0));
 b_sprint(STRCRLF);
 b_exit(0);
 /**/
}

/****************************/
int     test(mes,no,mul)
unsigned char   mes[32+1];
unsigned char   no[32+1];
double  mul;
{
      b_sprint(no);b_sprint(")");
      b_sprint(mes);
      b_fprint(b_pi(mul));b_sprint(STRCRLF);
 }

/****************************/
```

リスト3：コンパイルしても正しく動くプログラム

```
 10 /* こうすれば動く */
 20 str s
 30 test(" π  =",1,1#)                  /*ここと
 40 test(" π/2 =",2,0.5#)
 50 test(" 2π =",3,2#)                  /*ここと
 60 print "----------hit key----------";
 70 s=inkey$
 80 print
 90 end
100 /*
110 func test(mes;str,no;int,mul;float) /*ここ
120     print no;")";
130     print mes;
140     print pi(mul)
150 endfunc
```

の分岐先を表すために使われる。X-BASICではgosubやgotoの飛び先には行番号を使うけれど，Cでは行番号の代わりに名前（ラベル名）を使う。そこでBCは適当な名前をでっちあげるというわけだ。この説明を読んでのとおり，gosubやgotoを使わなければSやLから始まる名前を使っても構わない。

b_で始まる名前が使えないのには次のようなわけがある。X-BASICとCでは同じ名前で同じような機能の関数がいくつもあって，名前も機能もまったく同じなら問題はないんだけど，返り値なんかが少しずつ違っている場合がある。たとえば，fopenという関数はX-BASICでもCでもファイルをオープンするという働きは同じなのに，返す値がX-BASICではファイル番号で，CではFILE構造体へのポインタというものになる。だからCのfopenをそのまま使うことができないんだ。

そこでBCではb_fopenという別の関数を用意して，fopenが出てきたらすかさずb_をくっつけて，この関数の呼び出しに変更する。これで安心してfopenが使えるようになる代わりにb_fopenという名前を別の目的で使うことができなくなるわけだ。

$で終わる名前を使うときの注意点というのはb_の場合に少し似ている。Cでは$の文字は名前に使うことができないので，BCは$を大文字のSに置き換える。ユーザーズマニュアルに載っている例をそのまま使うと，bun$はbunSにコンバートされる。だから，bun$を使っているときにbunSという名前が別にあるとダブってしまうということだ。

strtmp0といった名前もBCが勝手に使う名前で，文字列操作を行うときに使われる。Cでは文字変数がなく，char型の配列を文字列とみなして使うのだった。そのため文字列操作は手間が掛かる，というところまでは話したよね。ここで，

```
100 str s="ABCDEFG"
110 s=right$(s,1)+left$(s,1)
```

というプログラムをCにコンバートしたときのことを考えよう。

BCによってsはchar型の配列に変換されている。すると，上のプログラムは，配列の最後の要素と最初の要素だけを取り出して，元の配列に収め直すものとなる。これを頭から順に処理しようとすると最後の要素を最初に持ってきて，それから最初の要素を2番目に持ってくるという形になるだろう。よーく考えてみると，これでは困るんだ。“GA”となるはずが“GG”になってしまう。なぜって，最後の要素を先頭に持ってきた時点で，今まで最後にあった要素が先頭の要素になってしまうからだ。

これを避ける手っ取り早い方法は，もうひとつ別の配列を用意して，それ上で作業を行い，終わってから一括してsに戻すことだ。この「別の配列」にあたるのがstrtmpというわけ。まどろっこしい説明だけど，わかってもらえただろうか？

石の都合ということもある

そのほかにもマニュアルには「してはいけないこと」がいくつか載っているが，そのほとんどが

・endfuncのあとにmain部のプログラムを続けて書いてはいけない
・funcで宣言した関数のなかから関数の外にgoto文で飛び出してはいけない

といったあたりまえのことなので説明するのはやめておく。おっと，ひとつだけ重要なことがあるので，こいつについて話しておこう。マニュアルには，

・fumc内で大きな配列を確保してはいけません。大きな配列はメインルーチンで確保してください。

とある。

この理由を説明するにはMC68000の話をしなければならない。が，そんな話を聞いてもあまり喜ぶ人もいないだろうから，ごく簡単に（はしょって）説明する。

C言語では（動的な）局所変数はある場所（スタック上）に作られる。68000というMPUにはこの「場所を確保する命令」があって，XCはその命令を使って局所変数を置いておく場所を作る。ところが，この命令で確保される領域は一定の大きさ（32768バイト）までに制限されているのだ。うまくやればこれ以上の大きさを確保する手段もあるけれど，XCはそこまではやってくれない。また，スタック領域の大きさにも限りがあって，むやみに使うわけにもいかないのだよ。

こういった理由で局所的な配列は「あまり」大きく取ることができないわけ。メインルーチンで宣言する配列の大きさにはこのような制限はないし，万一，宣言した配列が大きすぎてメモリが足りなければプログラムの起動時にエラーが出るはずだから安全なんだ。

これ，よろしくお願いね

さて，大詰めだ。最後にBCの大バグを公表しよう。次のようにプログラムを書いたとする。

〈例4〉

```
 100 test("ABCDEFG",10)
 110 end
1000 func test(s;str,n)
1010 while n>0
1020     print s
1030     n=n-1
1040 endwhile
1050 endfunc
```

ここで1000行以下の関数はforを使っていないのがちょっとわざとらしいが，nの値をだんだん減らしていって0になるまで文字列sを表示する，結局sをn回表示する関数ということになる。で，このプログラムをコンパイルして実行してみると，なんと！バスエラーを起こして止まってしまうんだ。さあ困った。いったいどうなっているんだ？悪いことはしていないはずなのに！

この症状はX-BASICとBCの仕様の違いというか，はっきりいってBCのバグといってよいだろう。X-BASICでは関数の仮引数の型が省略されたときはintとみなすが，BCは「直前の引数と同じ型」として処理してしまうのだ。上の例だとnがstr型になってしまったわけ。

最初に書いたように，仮引数と実引数の型が違ってもCコンパイラは文句をいわない。そのため，平気な顔でchar型の配列と数字の0を比較しようとして（1010行）死んでしまうことになる。こうならないようにするためには面倒でも，

```
1000 func test(s;str,n;int)
```

のようにint型の仮引数もちゃんと宣言するようにしなければならない。くれぐれも，シャープさんにはBCのバージョンアップをお願いしたい（オプティマイザもよろしくね）。

X-BASIC+BCでプログラムを作るうえでの注意点はだいたいこんなものだ。こうやって書き並べてみると結構な量になってびっくりしている。けれど，きちんと文法に従い，BCのマニュアルを読んでいれば，ここに書いたようなことはそうそう起こるものではないと思う。

そうそう，マニュアルといえば，XCの「あの」ぶ厚いマニュアルのうちBCに関係あるところはほんの数ページしかない。「だから，説明不足なんじゃないか！」と怒っている人もいるかもしれないけど，逆にいえば，たったそれっぽっちなんだもの，ちゃんと読んでおいてほしいな。

と，教訓めいた終わり方をするのは気がひけるけど，やっぱりこれでおしまいにしよう。ちゃんちゃん。

特別講義2 Cでアセンブリ言語の勉強を

Nakamori Akira
中森　章

いかにCが魅力的であっても，パソコン，特に8ビットマシンではその力を発揮できないと考える方も多いでしょう。そして，やっぱりマシン語にいちばん興味があるという人，このユニークな中森氏の講義をぜひとも読んでみてください。

C言語とアセンブリ言語

C言語はアセンブリ言語の代用品としてよく利用されますが，しょせんはコンパイラ言語ですから，アセンブリ言語に比べて効率の悪いコード（たいていはアセンブリ言語のプログラム）を出力してしまいます。確かにこれは本当のことで，アセンブリ言語一本槍のプログラマの論理のより所となっています。

しかし，実は必ずしもそうとは言えない場合もあります。最近ではCコンパイラの最適化の技術が進み，いかにすごい最適化が行われているかがCコンパイラの「売り」のひとつになっています。そして，プログラムのある局面の記述については，Cコンパイラはベテランのプログラマがアセンブリ言語で記述したコードとまったく同じか，時にはそれよりも効率のよいコードを出力します。ですから，アセンブリ言語を覚えたばかりの人がCコンパイラの出力よりも効率のよいコードを書くことはほとんど不可能です。

また，ベテランのプログラマにとっても，C言語の出力するコードをあとから手でさらに最適化することによって，自分の能力以上のコードを作ることも可能になります。しかし，多くの場合はCコンパイラの出力するコードで満足（我慢？）するプログラマが多いようです。もっとも，C言語の記述がどのようなコードに変換されるかを知っておけば，ある程度自分の思いどおりの（効率よい）コードをCコンパイラに出力させることができますから，あえて出力コードに手を入れる必要はありませんが。

また，思いどおりの出力コードを得るという，大それたことを考えない場合でも，Cコンパイラの出力するコードには学ぶべきところが多くあります。特に，アセンブリ言語をこれから勉強しようという人にとっては，Cコンパイラの出力コードはよいテキストになるでしょう。Cコンパイラの出力するコードの中に現れないような命令は，アセンブリ言語だけでプログラミングを行う場合も，まず必要ありません。Cコンパイラの出力コードに使われている命令を完全に理解することができたら，あなたはもうアセンブリ言語上級者なのです。

というわけで，いくつかのC言語の記述に対して，それらがCコンパイラによってどのようなコードに変換されるのかということを，これから見ていくことにしましょう。コンパイラは，Z80代表としてαC,8086代表としてTurbo C，68000代表としてXCを用いますが，これら以外に特徴的なコードを出力するコンパイラがあればその都度紹介したいと思います。

なお，C言語の文法は特に説明はしませんが，第1部の入門編やAppendixのリファレンスを参照してくださいね。また，アルゴリズムの記述にはAlgol（もはや知っている人はいないかなあ）に近いオリジナル（適当な）言語を使用していますが，BASICがわかる人にはわかると思います。

ループ構造

最初はループ構造を見ていきましょう。C言語では，

for
do～while
while

という3種類のループ構造を使用することができます（goto文を用いたループは考えていない）。これらは，終了条件を判断する位置が微妙に異なっていて，それぞれ時と場合に応じて使い分けます。このようなループをアセンブリ言語で記述することは難しそうですが，実は案外単純に実現できて，Cコンパイラで出力されるコードもかなり似ているのです。

これらのループ構造の記述に対して，実際のCコンパイラがどのようなコードを出力するかをリスト1に示します。以下にループ構造に対するアルゴリズムを説明しますからそれと照らし合わせながら読んでみてください。

1）forループ

forループは，通常，有限回の繰り返しを記述します。たとえば，

for(i=0;i<100;i++)　文;

というループを考えましょう。意味からいえば，Cコンパイラのコードとしては，アルゴリズム1のようになるべきですが，コードを出力する都合からアルゴリズム2のようになっているコンパイラが多いようです。

アルゴリズム1にしても，アルゴリズム2にしても，0回ループ（ループしない）

```
アルゴリズム1　ループ
        i ← 0
LOOP:
        if i ≧ 100 then goto EXIT
        "文"の実行
        i ← i+1
        goto LOOP
EXIT:

アルゴリズム2　ループ
        i ← 0
        goto SKIP
LOOP:
        "文"の実行
        i ← i+1
SKIP:
        if i < 100 then goto LOOP

アルゴリズム3　ループ
        i ← 0
LOOP:
        "文"の実行
        i ← i+1
        if i < 100 then goto LOOP

アルゴリズム4　ループ
LOOP:
        "文"の実行
        i ← i-1
        if i ≠ 0 then goto LOOP

アルゴリズム5　ループ
LOOP:
        i ← i-1
        if i = 0 then goto EXIT
        "文"の実行
        goto LOOP
EXIT:

アルゴリズム6　ループ
        goto SKIP
LOOP:
        "文"の実行
SKIP:
        i ← i-1
        if i ≠ 0 then goto LOOP
```

の場合があることを考慮してループの最初で終了条件をチェックします（アルゴリズム2では，ループの最後で行うチェックにジャンプして1回目のチェックをしているのですが）。しかし今の場合，iの初期値が0で終了値が100（正確には99）ですから，必ず1回はループすることがわかっています。このため，アルゴリズム1や2では1回とはいえ確実に余分な終了条件のチェックを行うことになります。したがって，これを最適化したアルゴリズム3のようなコードを出力するコンパイラもかなりあります。

ただし，初期値が変数で与えられている場合はアルゴリズム1か2のようにならざるを得ません。また，アルゴリズム3を使うコンパイラでは，最初からループしないことがわかっている場合は，ループに対するコードをまったく出力しないものもあります。リスト1を見てもらえばわかりますが，αCはアルゴリズム1，Turbo Cはアルゴリズム2，XCはアルゴリズム3を採用しています。

2)　do～whileループ

do～whileループはループ回数が不定のときに用いられます。ループの最後で終了条件の判定を行いますから，必ず1回はループを行います。たとえば，

do 文 ; while(－j) ;

というループを考えましょう。このループに対する出力コードのアルゴリズムは，ループの最後で条件判定をするため，forループのアルゴリズム3と同様です。具体的にはアルゴリズム4のようになりますが，do～whileループに関してはほかのアルゴリズムはないようです。

しかし，終了条件が，

1＞2

というように常に成立しない場合は，終了条件のチェックを行うコードを出力せずに，次に進むということはあるかもしれませんが。

リスト 1

1.　ループ(C言語)

```
main()
{
        register int i;
        register int j;

        for(i=0;i<100;i++) j=i+10;

        do j=i+10; while(--j);

        do j=i+10; while(j--);

        while(--i) j=i+j;

        while(i--) j=i+j;
}
```

2.　ループ(αC)

```
;
;       このリストはαCのオブジェクトを
;       逆アセンブルして作成したものです
;
SDLI    EQU     190H            ; ローカル変数の取り出し
BLAS    EQU     1F9H            ; HLとDEの大小比較
;
_main:
;
        PUSH    BC
        LD      HL,0FFFCH
        ADD     HL,SP
        LD      SP,HL
        LD      B,H
        LD      C,L
;
        LD      H,B
        LD      L,C
        XOR     A
        LD      (HL),A          ; i=0
        INC     HL
        LD      (HL),A
L000:
;
        CALL    SDLI            ; HL <- i (BC+0)
        DB      0
        LD      DE,100
        CALL    BLAS
        JP      NC,L001         ; i>=100ならL001(おわり)
;
        CALL    SDLI            ; HL <- i (BC+0)
        DB      0
        LD      DE,10
        ADD     HL,DE           ; HL <- i+10
        EX      DE,HL           ; DE <- i+10
        LD      HL,2
        ADD     HL,BC           ; HL <- jのアドレス BC+2
        LD      (HL),E
        INC     HL
        LD      (HL),D          ; j <- DE
;
        LD      H,B
        LD      L,C             ; HL <- iのアドレス BC
        LD      E,(HL)
        INC     HL
        LD      D,(HL)          ; DE <- i
        INC     DE              ; DE <- i+1
        LD      (HL),D
        DEC     HL
        LD      (HL),E          ; i <- DE
        JP      L000
L001:
;
        CALL    SDLI            ; HL <- i (BC+0)
        DB      0
        LD      DE,10
        ADD     HL,DE           ; HL <- i+10
        EX      DE,HL           ; DE <- i+10
        LD      HL,2
        ADD     HL,BC           ; HL <- jのアドレス BC+2
        LD      (HL),E
        INC     HL
        LD      (HL),D          ; j <- DE
;
        LD      HL,2
        ADD     HL,BC           ; HL <- jのアドレス BC+2
        LD      E,(HL)
        INC     HL
        LD      D,(HL)          ; DE <- j
        DEC     DE              ; DE <- j-1
        LD      (HL),D
        DEC     HL
        LD      (HL),E          ; j <- DE
        LD      A,D
        OR      E
        JP      NZ,L001         ; DE(j)<>0ならL001
L002:
;
        CALL    SDLI            ; HL <- i (BC+0)
        DB      0
        LD      DE,10
        ADD     HL,DE           ; HL <- i+10
        EX      DE,HL           ; DE <- i+10
        LD      HL,2
        ADD     HL,BC           ; HL <- jのアドレス BC+2
        LD      (HL),E
        INC     HL
        LD      (HL),D          ; j <- HL
;
        LD      HL,2
        ADD     HL,BC           ; HL <- jのアドレス BC+2
        LD      E,(HL)
        INC     HL
        LD      D,(HL)          ; DE <- j
        DEC     DE              ; DE <- j-1
        LD      (HL),D
        DEC     HL
        LD      (HL),E          ; j <- DE
        INC     DE              ; DE <- DE+1 ひとつ前のj
        LD      A,D
        OR      E
        JP      NZ,L002         ; DE<>0ならL002
L003:
;
        LD      H,B
        LD      L,C             ; HL <- iのアドレス BC
        LD      E,(HL)
        INC     HL
        LD      D,(HL)          ; DE <- i
        DEC     DE              ; DE < i-1
        LD      (HL),D
        DEC     HL
        LD      (HL),E          ; i <- DE
        LD      A,D
        OR      E
        JP      Z,L004          ; DE(i)==0ならL004(おわり)
;
        CALL    SDLI            ; HL <- i (BC+0)
        DB      0
        PUSH    HL
        CALL    SDLI            ; HL <- j (BC+2)
        DB      2
        POP     DE              ; DE <- i
        ADD     HL,DE           ; HL <- i+j
        EX      DE,HL           ; DE <- i+j
        LD      HL,2
        ADD     HL,BC           ; HL <- jのアドレス BC+2
        LD      (HL),E
        INC     HL
        LD      (HL),D          ; j <- DE
        JP      L003
L004:
;
        LD      H,B
        LD      L,C             ; HL <- iのアドレス BC
        LD      E,(HL)
        INC     HL
        LD      D,(HL)          ; DE <- i
        DEC     DE              ; DE <- i-1
        LD      (HL),D
        DEC     HL
        LD      (HL),E          ; i <- DE
        INC     DE
        LD      A,D
        OR      E
        JP      Z,L005          ; DE(i)==0ならL005(おわり)
;
        CALL    SDLI            ; HL <- i (BC+0)
        DB      0
        PUSH    HL
        CALL    SDLI            ; HL <- j (BC+2)
        DB      2
        POP     DE              ; DE <- i
        ADD     HL,DE           ; HL <- i+j
        EX      DE,HL           ; DE <- i+j
        LD      HL,2
        ADD     HL,BC           ; HL <- jのアドレス BC+2
        LD      (HL),E
        INC     HL
        LD      (HL),D          ; j <- DE
        JP      L004
L005:
;
        EX      DE,HL
        LD      HL,4
        ADD     HL,SP
        LD      SP,HL
        EX      DE,HL
        POP     BC
        RET
```

3.　ループ(Turbo C)

```
;
;       このリストは Turbo C の出力リスト
;       にコメントを付け加えたものです
;
        name    cnt
_TEXT   segment byte public 'CODE'
DGROUP  group   _DATA,_BSS
        assume  cs:_TEXT,ds:DGROUP,ss:DGROUP
_TEXT   ends
_DATA   segment word public 'DATA'
_d@     label   byte
_DATA   ends
_BSS    segment word public 'BSS'
_b@     label   byte
_BSS    ends
_TEXT   segment byte public 'CODE'
; Line 2
_main   proc    near
        push    si
        push    di
; Line 3
; Line 4
; Line 5
; Line 6
        xor     si,si           ; si = i
        jmp     short @5        ; di = j
@4:
        mov     di,si
        add     di,10           ; j <- i+10
        inc     si              ; i++
@5:
        cmp     si,100
        jl      @4              ; i<100 ならループ
; Line 7
; Line 8
@8:
        mov     di,si
        add     di,10           ; j <- i+10
        dec     di              ; --j
        mov     ax,di
        or      ax,ax
        jne     @8              ; j<>0 ならループ
; Line 9
; Line 10
@11:
        mov     di,si
        add     di,10           ; j <- i+10
        mov     ax,di
        dec     di              ; j--
        or      ax,ax
        jne     @11             ; (j+1)<>0 ならループ
; Line 11
; Line 12
@12:
        dec     si              ; --i
        mov     ax,si
        or      ax,ax
        je      @14             ; i<>0 ならおわり
        mov     ax,si
        add     ax,di
        mov     di,ax           ; j <- i+j
        jmp     short @12       ; ループ
; Line 13
; Line 14
@14:
        mov     ax,si
        dec     si              ; i--
        or      ax,ax
        je      @1              ; (i+1)==0 ならおわり
        mov     ax,si
        add     ax,di
        mov     di,ax           ; j <- i+j
        jmp     short @14       ; ループ
; Line 15
@1:
        pop     di
        pop     si
        ret
_main   endp
_TEXT   ends
_DATA   segment word public 'DATA'
_s@     label   byte
_DATA   ends
_TEXT   segment byte public 'CODE'
        public  _main
_TEXT   ends
        end
```

4.　ループ(XC)

```
;
;       このリストは XC の出力リストに
;       コメントを付け加えたものです
;
        include fefunc.h
        .XREF   __main
        .XDEF   _main
        .TEXT
_main:
__main:
        BRA     L1              ; D7 = i  D6 = j
L20001:
        MOVE.L  D7,D6
        ADD.L   #10,D6          ; j <- i+10
L6:
        ADDQ.L  #1,D7           ; i++
L4:
        MOVE.L  D7,D0
        CMP.L   #100,D0
        BLT     L20001          ; i<100 ならループ
L9:
        MOVE.L  D7,D6
        ADD.L   #10,D6          ; j <- i+10
L7:
        SUBQ.L  #1,D6           ; --j
        BNE     L9              ; j<>0 ならループ
L8:
L12:
        MOVE.L  D7,D6
        ADD.L   #10,D6          ; j <- i+10
L10:
        MOVE.L  D6,D0
        SUBQ.L  #1,D6           ; j--
        TST.L   D0
        BNE     L12             ; (j+1)<>0 ならループ
L11:
        BRA     L13
L20003:
        MOVE.L  D7,D0
        ADD.L   D6,D0
        MOVE.L  D0,D6           ; j <- i+j
L13:
        SUBQ.L  #1,D7           ; --i
        BNE     L20003          ; i<>0 ならループ
        BRA     L15
L20005:
        MOVE.L  D7,D0
        ADD.L   D6,D0
        MOVE.L  D0,D6           ; j <- i+j
L15:
        MOVE.L  D7,D0
        SUBQ.L  #1,D7           ; i--
        TST.L   D0
        BNE     L20005          ; (i+1)<>0 ならループ
L3:
        MOVEM.L (SP)+,D6/D7
        UNLK    A6
        RTS
L1:
        LINK    A6,#0
        MOVEM.L D6/D7,-(SP)
        CLR.L   D7
        BRA     L4
        .DATA
        .EVEN
        .END
```

3)　whileループ

whileループもループ回数が不定のときに用いられます。繰り返し動作の最初で終了条件の判定を行いますから，ループを行わない場合もあります。たとえば，

while(−−j) 文；

という文に対するコードは，意味どおりにはアルゴリズム5によって作られます。しかし，コンパイラの都合でアルゴリズム6のようになる場合もあります。これはforループのアルゴリズム1と2の関係に似ていますね。例によって，必ず成立しないループ条件が指定された場合は，whileループに対するコードをまったく出さないコンパイラがあるかもしれません。なお，リスト1から，αCはアルゴリズム1，Turbo Cもアルゴリズム1，XCはアルゴリズム2を採用していることがわかります。

switch文

switch文は変数の値に対応して，実行する動作を選択する場合に用います。まずはswitch文に対してコンパイラの出力するコードをリスト2に示しましょう。それではまず，

```
switch(i) {
case 100：文1；break；
case 200：文2；break；
case 300：文3；break；
case 400：文4；break；
default：文5；
}
```

というswitch文を考えましょう。この文は，いうまでもなく変数iの値が100である場合は"文1"を実行し，200である場合は"文2"を実行し，300である場合は"文3"を実行し，400である場合は"文4"を実行し，それ以外の場合は"文5"を実行することを指示します。

「なんだ，if文を並べれば同じことができるじゃないか」という人がいると思いますが，実はまったくそのとおりなのです。それでは，if文とどこが違うのかといえば，switch文のほうがプログラムを見やすく記述できるという程度のことでしかありません。

実際Cコンパイラも，基本的には，if文を立て続けに並べたのと同じコードを出力します。つまり，アルゴリズム7です。αCやXCはこのアルゴリズムでコードを作っています。ただし，アルゴリズム7はコードサイズが大きくなる傾向にあるので，アルゴリズム8のようにコードサイズを小さくするものも考えられています。これはif文を並べて書く代わりに，ひとつのif文をCASEの個数回だけループさせせようという方針です。このとき，比較する値はメモリ内に置かれ，これらの値は配列要素をアクセスする要領でアクセスされ，変数と値が一致したときにif文のループを抜け出します。

アルゴリズム8はなかなか技巧的で，読む人をうならせるものがあります。しかし，メモリアクセスを何度もする点，条件分岐を使用する点を考慮すると，実行速度はif文を立て続けに並べるよりもかなり遅くなりそうです。Turbo Cがこのアルゴリズムでコードを出力しますが，コンパイラとしては，出してほしくないコードです。

さて，これまで示したように，Cコンパイラはcaseの値が飛び飛びの場合はif文を使ってswitch文を実現しますが，CASEの値が1ずつ変化している場合は別のアルゴリズムが用いられることが多いようです。たとえば，

```
switch(i) {
case 1：文1；break；
case 2：文2；break；
case 3：文3；break；
case 4：文4；break；
default：文5；
}
```

というswitch文に対してはアルゴリズム9が用いられます。

これは，変数iを直接アドレスに対応させ，そのアドレスで示されるメモリ内にジャンプ先のアドレスを格納しておくという，かなり技巧的なアルゴリズムです。このアルゴリズムの素晴しいところは，メモリを1回アクセスするだけでジャンプ先を得る（メモリ間接ジャンプですよ）ことができる点にあります。アルゴリズム8の後半でも同じことをしていますから，比べてみると面白いでしょう（アルゴリズム8の欠点はif文をループさせる点のみであり，後半のメモリ間接ジャンプは見事なものです）。

このアルゴリズム9はCASEの値が1ずつ増加しているときにしか使用できませんが，ある程度なら値が飛び飛びになっていても使用することができます。それは，メモリ内に格納するジャンプ先にDEFAULT処理を行う場合のアドレスを挿入することによって行います。たとえば，

```
switch(i) {
case 1：文1；break；
case 3：文2；break；
case 6：文3；break；
```

リスト 2

1. switch(C言語)

```
int i=0;

main()
{
        switch(i){
        case 100: i=100; break;
        case 200: i=200; break;
        case 300: i=300; break;
        case 400: i=400; break;
        default:  i=500;
        }

        switch(i){
        case 1:   i=101; break;
        case 2:   i=201; break;
        case 3:   i=301; break;
        case 4:   i=401; break;
        default:  i=501;
        }

        switch(i){
        case 1:   i=102; break;
        case 3:   i=302; break;
        case 7:   i=702; break;
        case 9:   i=902; break;
        default:  i=802;
        }
}
```

2. switch(αC)

```
;
;       このリストはαCのオブジェクトを
;       逆アセンブルして作成したものです
;
;
;       変数のベースアドレスを1000Hとした場合
;
_i      EQU     1000H
;
_main:
        PUSH    BC
;
        LD      HL,(_i)
        LD      A,100
        CP      L               ; 下位8ビットの比較
        JP      NZ,,L000        ; 一致しなければ次へ
        LD      A,0
        CP      H               ; 上位8ビットの比較
        JP      Z,C000
;
L000:
        LD      A,200
        CP      L               ; 下位8ビットの比較
        JP      NZ,,L001        ; 一致しなければ次へ
        LD      A,0
        CP      H               ; 上位8ビットの比較
        JP      Z,C001
;
L001:
        LD      A,44
        CP      L               ; 下位8ビットの比較
        JP      NZ,,L002        ; 一致しなければ次へ
        LD      A,256
        CP      H               ; 上位8ビットの比較
        JP      Z,C002
;
L002:
        LD      A,144
        CP      L               ; 下位8ビットの比較
        JP      NZ,,L003        ; 一致しなければ次へ
        LD      A,256
        CP      H               ; 上位8ビットの比較
        JP      Z,C003
;
L003:
        JP      C004
;
C000:
        LD      HL,100
        LD      (_i),HL         ; i <- 100
        JP      L004
C001:
        LD      HL,200
        LD      (_i),HL         ; i <- 200
        JP      L004
C002:
        LD      HL,300
        LD      (_i),HL         ; i <- 300
        JP      L004
C003:
        LD      HL,400
        LD      (_i),HL         ; i <- 400
        JP      L004
C004:
        LD      HL,500
        LD      (_i),HL         ; i <- 500
;
L004:
        LD      HL,(_i)
        LD      A,1
        CP      L               ; 下位8ビットの比較
        JP      NZ,,L005        ; 一致しなければ次へ
        LD      A,0
        CP      H               ; 上位8ビットの比較
        JP      Z,C005
;
L005:
        LD      A,2
        CP      L               ; 下位8ビットの比較
        JP      NZ,,L006        ; 一致しなければ次へ
        LD      A,0
        CP      H               ; 上位8ビットの比較
        JP      Z,C006
;
L006:
        LD      A,3
        CP      L               ; 下位8ビットの比較
        JP      NZ,,L007        ; 一致しなければ次へ
        LD      A,0
        CP      H               ; 上位8ビットの比較
        JP      Z,C007
;
L007:
        LD      A,4
        CP      L               ; 下位8ビットの比較
        JP      NZ,,L008        ; 一致しなければ次へ
        LD      A,0
        CP      H               ; 上位8ビットの比較
        JP      Z,C008
;
L008:
        JP      C009
;
C005:
        LD      HL,101
        LD      (_i),HL         ; i <- 101
        JP      L009
C006:
        LD      HL,201
        LD      (_i),HL         ; i <- 201
        JP      L009
C007:
        LD      HL,301
        LD      (_i),HL         ; i <- 301
        JP      L009
C008:
```

default：文4；

｝

のように，CASEの値の増加が1ずつでない場合でも，アルゴリズム9のLABEL以下に格納してあるジャンプ先アドレスを，

LABEL：

DW　CASE1のアドレス
DW　DEFAULTのアドレス
DW　CASE3のアドレス
DW　DEFAULTのアドレス
DW　DEFAULTのアドレス
DW　CASE6のアドレス

とすることで適用できるのです。

もちろん，最初に示したようなCASEの値がバラバラの場合も，

LABEL：

DW　CASE100のアドレス
DW　DEFAULTのアドレス
……………………………
DW　DEFAULTのアドレス
DW　CASE200のアドレス
DW　DEFAULTのアドレス
……………………………
DW　DEFAULTのアドレス
DW　CASE300のアドレス
DW　DEFAULTのアドレス
……………………………

という具合に，ジャンプ先をひたすら並べ続けることによって，アルゴリズム9を適用できますが，この場合のコードサイズは膨大なものとなってしまいます。このため，アルゴリズム9はCASEの値の差がある程度小さくないと有効ではありません。しかし，現実は差がある範囲に収まる場合が多いようで，アルゴリズム9は16ビット以上のCPUのCコンパイラではほとんど常識になっているようです。

リスト2を見てわかるように，CASEの値がある程度近い場合は，Turbo CもXCもこのアルゴリズムになっていますね。もちろん，このアルゴリズム9は8ビットCPUでも可能なはずですが，αCだけでなく，8ビット最高速といわれているLSI-Cにおいてもなぜか使用されていません。どうしてなんでしょう。

ところで，アルゴリズム9はメモリ内にジャンプ先のアドレスを置いたわけですが，さらに技巧的な方法としては，アドレスではなく，メモリ間接ジャンプがあるアドレスからのオフセット値を置く方法もあります。

あるアドレスからその近くのアドレスまでのオフセット値（アドレスの値の差）は，通常，アドレスを表すビット数（たとえば

```
        LD      HL,401
        LD      (_i),HL         ; i <- 401
        JP      L009
C009:
        LD      HL,501
        LD      (_i),HL         ; i <- 501
;
L009:
        LD      HL,(_i)
        LD      A,1
        CP      L               ; 下位8ビットの比較
        JP      NZ,,L010        ; 一致しなければ次へ
        LD      A,0
        CP      H               ; 上位8ビットの比較
        JP      Z,C010
;
L010:
        LD      A,3
        CP      L               ; 下位8ビットの比較
        JP      NZ,,L011        ; 一致しなければ次へ
        LD      A,0
        CP      H               ; 上位8ビットの比較
        JP      Z,C011
;
L011:
        LD      A,7
        CP      L               ; 下位8ビットの比較
        JP      NZ,,L012        ; 一致しなければ次へ
        LD      A,0
        CP      H               ; 上位8ビットの比較
        JP      Z,C012
;
L012:
        LD      A,9
        CP      L               ; 下位8ビットの比較
        JP      NZ,,L013        ; 一致しなければ次へ
        LD      A,0
        CP      H               ; 上位8ビットの比較
        JP      Z,C013
;
L013:
        JP      C014
;
C010:
        LD      HL,102
        LD      (_i),HL         ; i <- 102
        JP      L014
C011:
        LD      HL,302
        LD      (_i),HL         ; i <- 302
        JP      L014
C012:
        LD      HL,702
        LD      (_i),HL         ; i <- 702
        JP      L014
C013:
        LD      HL,902
        LD      (_i),HL         ; i <- 902
        JP      L014
C014:
        LD      HL,802
        LD      (_i),HL         ; i <- 801
;
L014:
        POP     BC
        RET
```

3.　switch(Turbo C)

```
;
;       このリストは Turbo C の出力リスト
;       にコメントを付け加えたものです
;
        name    sw
_TEXT   segment byte public 'CODE'
DGROUP  group   _BSS
        assume  cs:_TEXT,ds:DGROUP,ss:DGROUP
_TEXT   ends
_DATA   segment word public 'DATA'
_d@     label   byte
_DATA   ends
_BSS    segment word public 'BSS'
_b@     label   byte
_BSS    ends
_DATA   segment word public 'DATA'
        public  _i
_i      label   word
        dw      0
_DATA   ends
_TEXT   segment byte public 'CODE'
; Line 4
_main   proc    near
; Line 5
        mov     ax,word ptr DGROUP:_i
        mov     cx,4                    ; 4回ループ
        mov     bx,offset @20
@22:
        cmp     cs:[bx],ax              ; 等しい値があれば
        je      @21                     ; おしまい
        inc     bx                      ; ポインタ更新
        inc     bx
        loop    @22                     ; ループ
        jmp     short @7                ; 等しいものがなかった
@21:
        jmp     word ptr cs:[bx+8]      ; それぞれのケースの処理へ
@20:
        dw      100                     ; case 100
        dw      200                     ; case 200
        dw      300                     ; case 300
        dw      400                     ; case 400
        dw      @3                      ; case 100 の処理アドレス
        dw      @4                      ; case 200 の処理アドレス
        dw      @5                      ; case 300 の処理アドレス
        dw      @6                      ; case 400 の処理アドレス
; Line 6
@3:
        mov     word ptr DGROUP:_i,100  ; case 100
        jmp     short @2
; Line 7
@4:
        mov     word ptr DGROUP:_i,200  ; case 200
        jmp     short @2
; Line 8
@5:
        mov     word ptr DGROUP:_i,300  ; case 300
        jmp     short @2
; Line 9
@6:
        mov     word ptr DGROUP:_i,400  ; case 400
        jmp     short @2
; Line 10
@7:
        mov     word ptr DGROUP:_i,500  ; default
; Line 11
@2:
; Line 12
; Line 13
        mov     ax,word ptr DGROUP:_i
        dec     ax
        cmp     ax,3
        ja      @13                     ; (i-1)>3 なら default
        mov     bx,ax
        shl     bx,1                    ; オフセット値を計算
        jmp     word ptr cs:@23[bx]     ; メモリ間接ジャンプ
@23     label   word
        dw      @9                      ; case 1
        dw      @10                     ; case 2
        dw      @11                     ; case 3
        dw      @12                     ; case 4
; Line 14
@9:
        mov     word ptr DGROUP:_i,101  ; case 1
        jmp     short @8
; Line 15
@10:
        mov     word ptr DGROUP:_i,201  ; case 2
        jmp     short @8
; Line 16
@11:
        mov     word ptr DGROUP:_i,301  ; case 3
        jmp     short @8
; Line 17
@12:
        mov     word ptr DGROUP:_i,401  ; case 4
        jmp     short @8
; Line 18
@13:
        mov     word ptr DGROUP:_i,501  ; default
; Line 19
@8:
; Line 20
; Line 21
        mov     ax,word ptr DGROUP:_i
        dec     ax
        cmp     ax,8
        ja      @19                     ; (i-1)>8 なら default
        mov     bx,ax
        shl     bx,1                    ; オフセット値を計算
        jmp     word ptr cs:@24[bx]     ; メモリ間接ジャンプ
@24     label   word
        dw      @15                     ; case 1
        dw      @19                     ; default
        dw      @16                     ; case 3
        dw      @19                     ; default
        dw      @19                     ; default
        dw      @19                     ; default
        dw      @17                     ; case 7
        dw      @19                     ; default
        dw      @18                     ; case 9
; Line 22
@15:
        mov     word ptr DGROUP:_i,102  ; case 1
        jmp     short @1
; Line 23
@16:
        mov     word ptr DGROUP:_i,302  ; case 3
        jmp     short @1
; Line 24
@17:
        mov     word ptr DGROUP:_i,702  ; case 7
        jmp     short @1
; Line 25
@18:
        mov     word ptr DGROUP:_i,902  ; case 9
        jmp     short @1
; Line 26
@19:
        mov     word ptr DGROUP:_i,802  ; default
; Line 27
; Line 28
@1:
        ret
_main   endp
_TEXT   ends
_DATA   segment word public 'DATA'
_s@     label   byte
_DATA   ends
_TEXT   segment byte public 'CODE'
        public  _main
_TEXT   ends
        end
```

4.　switch(XC)

```
;
;       このリストは XC の出力リストに
;       コメントを付け加えたものです
;
        include fefunc.h
        .GLOBL  _i
        .DATA
_i:
        .DC.L   0
        .XREF   __main
        .XDEF   _main
        .TEXT
_main:
__main:
        BRA     L1
L6:
        MOVE.L  #100,_i         ; case 100 の処理
        BRA     L4
L7:
        MOVE.L  #200,_i         ; case 200 の処理
        BRA     L4
L8:
        MOVE.L  #300,_i         ; case 300 の処理
        BRA     L4
L9:
        MOVE.L  #400,_i         ; case 400 の処理
        BRA     L4
L13:
        MOVE.L  #101,_i         ; case 1 (第2文) の処理
        BRA     L11
L14:
        MOVE.L  #201,_i         ; case 2 (第2文) の処理
        BRA     L11
L15:
        MOVE.L  #301,_i         ; case 3 (第2文) の処理
        BRA     L11
L16:
        MOVE.L  #401,_i         ; case 4 (第2文) の処理
        BRA     L11
L17:
        MOVE.L  #501,_i         ; default (第2文) の処理
        BRA     L11
L10002:
        .DC.L   L13             ; case 1 (第2文)
        .DC.L   L14             ; case 2 (第2文)
        .DC.L   L15             ; case 3 (第2文)
        .DC.L   L16             ; case 4 (第2文)
;
;       第3文
;
L11:
        MOVE.L  _i,D0
        SUB.L   #1,D0
        CMPI.L  #8,D0
        BHI     L24             ; (i-1)>8 なら default (第3文)
        ASL.L   #2,D0           ; オフセットの計算
        LEA     L10004(PC),A0   ; L10004 のアドレスを A0 に
        MOVE.L  (A0,D0.L),A0    ; オフセットを加算
        JMP     (A0)            ; メモリ間接ジャンプ
L20:
        MOVE.L  #102,_i         ; case 1 (第3文) の処理
        BRA     L18
L21:
        MOVE.L  #302,_i         ; case 3 (第3文) の処理
        BRA     L18
L22:
        MOVE.L  #702,_i         ; case 7 (第3文) の処理
        BRA     L18
L23:
        MOVE.L  #902,_i         ; case 9 (第3文) の処理
        BRA     L18
L24:
        MOVE.L  #802,_i         ; default (第3文) の処理
        BRA     L18
L10004:
        .DC.L   L20             ; case 1  (第3文)
        .DC.L   L24             ; default (第3文)
        .DC.L   L21             ; case 3  (第3文)
        .DC.L   L24             ; default (第3文)
        .DC.L   L24             ; default (第3文)
        .DC.L   L24             ; default (第3文)
        .DC.L   L22             ; case 7  (第3文)
        .DC.L   L24             ; default (第3文)
        .DC.L   L23             ; case 9  (第3文)
L18:
L3:
;
;       プログラムのおわり
;
        UNLK    A6
        RTS
L1:
        LINK    A6,#0
;
;       第1文
;
        MOVE.L  _i,D0
        CMPI.L  #100,D0
        BEQ     L6              ; i==100 なら case 100
        CMPI.L  #200,D0
        BEQ     L7              ; i==200 なら case 200
        CMPI.L  #300,D0
        BEQ     L8              ; i==300 なら case 300
        CMPI.L  #400,D0
        BEQ     L9              ; i==400 なら case 400
        MOVE.L  #500,_i         ; default (第1文) 処理
L4:
;
;       第2文
;
        MOVE.L  _i,D0
        SUB.L   #1,D0
        CMPI.L  #3,D0
        BHI     L17             ; (i-1)>3 なら default (第2文)
        ASL.L   #2,D0           ; オフセット計算
        LEA     L10002(PC),A0   ; L10002 のアドレスを A0 に
        MOVE.L  (A0,D0.L),A0    ; オフセットを加算
        JMP     (A0)            ; メモリ間接ジャンプ
        .DATA
        .EVEN
        .END
```

32ビット）よりも少ないビット数（たとえば16ビット）で表すことができます。このため，オフセット値をメモリから取り出す場合は，アドレスを取り出す場合よりも少ない回数のバスサイクルで行うことができ，命令の実行が高速になります。

これはSun-3のUNIX上のCコンパイラで行われているアルゴリズムですが，本当に最適化をするためにはバスサイクルの回数などということも考慮しなければならないのですね。具体的なアルゴリズムは，アルゴリズム9のちょっとした変形ですから各自考えてみましょう。Z80や8086の命令では実現（この場合は8ビットのオフセットを使うことになる）が難しいかもしれませんが，68000なら簡単だと思います。

定数の乗算

C言語では式の計算に掛け算や割り算を使うことができます(当たり前！)。しかし，たいていの8ビットCPUは掛け算命令や割り算命令を持っていません。また，掛け算命令や割り算命令があるCPUでも，それらの命令は実行速度が遅いため，特殊な場合はなんらかの工夫を行っています。ここでは，定数値との掛け算に対してCコンパイラがどのようなコードを出力するのかを見てみましょう。

まず，掛け算のプログラムとCコンパイラの出力をリスト3に示します。掛け算を行う場合に通常行われるのは掛け算用のサブルーチンを呼ぶことです。しかし，ある特定の場合は，サブルーチンを呼ぶまでもなく結果がわかってしまうのです。以下にいくつかの特定な場合について考えてみます。

1)　1倍

これは変数の値をそのまま転送するだけです。これをアルゴリズム10としましょう(大袈裟)。しかし，コンパイラによっては素直に1倍するタコもあるようです。

2)　2倍

これには，そのまま掛け算してしまうタコを除いて，2通りのアルゴリズムがあります。それは，シフトを利用する方法（アルゴリズム11）と足し算を利用する方法(アルゴリズム12）です。実行速度の点から見れば，足し算のほうが速いことは明らかなのですが，多くのコンパイラはシフトによる計算を行っているようです。ところで，整数値を1ビット左シフトすると，値が2倍になることは知ってますよねえ。

3)　3倍

2倍と1倍（自分自身）の加算によって行われます。これがアルゴリズム13です。2倍についてはアルゴリズム11と12のどちらかが用いられますが，コンパイラによっては，2倍のときにアルゴリズム11（シフト）を用いていながら，3倍のときの2倍にはアルゴリズム12を用いている変わり者もあります。

4)　4倍

圧倒的に自分自身を2ビット左シフトする（アルゴリズム14）コンパイラが多いようです。しかし，自分自身を2度加算しても4倍を求めることができます（アルゴリズム15)。8086のCコンパイラは1ビットシフトを2度繰り返して2ビットシフトを実現（アルゴリズム14）していますが，2度演算を行うならば，加算を2度行ったほう(アルゴリズム15)が高速のはずです。不思議ですね。実際，Z80のコンパイラではアルゴリズム15のほうがよく使われているようです。

ところで，2倍，4倍のアルゴリズムを見ればわかるように，2^n倍（2倍，4倍，8倍，16倍，……）を求めるときには，nビットの左シフトか，n回の加算が用いられることがわかりますね。

5)　5倍

さすがに5倍になると挫折(？)して，遅い掛け算命令を使用したり，遅い掛け算用サブルーチンを呼ぶコンパイラが多いよう

アルゴリズム7　switch

```
        if i = 100 then goto CASE100
        if i = 200 then goto CASE200
        if i = 300 then goto CASE300
        if i = 400 then goto CASE400
        "文5"の実行
        goto NEXT
CASE100:
        "文1"の実行
        goto NEXT
CASE200:
        "文2"の実行
        goto NEXT
CASE300:
        "文3"の実行
        goto NEXT
CASE400:
        "文4"の実行
NEXT:
```

アルゴリズム8　switch

```
        cnt ← CASEの個数
        tmp ← LABELのｱﾄﾞﾚｽ
LOOP:
        if i = *tmp then goto NEXT
        tmp ← tmp + ﾜｰﾄﾞｻｲｽﾞ
        cnt ← cnt - 1
        if cnt ≠ 0 then goto LOOP
        goto DEFAULT
LABEL:
        DW      100
        DW      200
        DW      300
        DW      400
        DW      CASE100のｱﾄﾞﾚｽ
        DW      CASE200のｱﾄﾞﾚｽ
        DW      CASE300のｱﾄﾞﾚｽ
        DW      CASE400のｱﾄﾞﾚｽ
CASE100:
        "文1"の実行
        goto EXIT
CASE200:
        "文2"の実行
        goto EXIT
CASE300:
        "文3"の実行
        goto EXIT
CASE400:
        "文4"の実行
        goto EXIT
NEXT:
        disp ← CASEの個数 × ﾜｰﾄﾞｻｲｽﾞ
        goto  *(tmp + disp)
DEFAULT:
        "文5"の実行
EXIT:
```

注）＊tmpはtmpの中に入っているアドレスで示されるメモリの内容（今は，100, 200, 300, 400のどれか）を示す。また，＊(tmp＋disp)はtmp＋dispで計算されるアドレスで示されるメモリの内容(今は，CASE100のアドレス，CASE200のアドレス，CASE300のアドレス，CASE400のアドレスのどれか）を示す。

アルゴリズム9　switch

```
        tmp ← i
        if tmp > CASEの最大値
                then goto DEFAULT
        tmp ← tmp - CASEの最小値
        tmp2 ← LABELのｱﾄﾞﾚｽ
        goto *(tmp2 + tmp × ﾜｰﾄﾞｻｲｽﾞ)
LABEL:
        DW      CASE1のｱﾄﾞﾚｽ
        DW      CASE2のｱﾄﾞﾚｽ
        DW      CASE3のｱﾄﾞﾚｽ
        DW      CASE4のｱﾄﾞﾚｽ
CASE1:
        "文1"の実行
        goto NEXT
CASE2:
        "文2"の実行
        goto NEXT
CASE3:
        "文3"の実行
        goto NEXT
CASE4:
        "文4"の実行
        goto NEXT
DEFAULT:
        "文5"の実行
NEXT:
```

注）＊(tmp2＋tmp×ワードサイズ)はtmp2＋tmp×ワードサイズで計算されるアドレスで示されるメモリの内容（今は，CASE1のアドレス，CASE2のアドレス，CASE3のアドレス，CASE4のアドレスのどれか）を示す。

です。しかし，まだ諦めないコンパイラもいます。確かに，4倍が計算できるコンパイラなら，それに自分自身を加えれば5倍になる（アルゴリズム16）ので，もう少し辛抱して5倍くらいは計算してもらいたいものです。例として用いているαC，Turbo C，XCの各コンパイラはすべて挫折していますが，LSI-C，MS-C，Sun-3のUNIX上のCコンパイラはアルゴリズム16で頑張っています。以後に6倍，7倍の場合を示してありますが，そこで用いられているアルゴリズムもこれらのコンパイラで使用されているものです。

6)　6倍

5倍が計算できるコンパイラは6倍も諦めません。計算の仕方としては2つに大別できます。すなわち，2倍と4倍を加えるもの（アルゴリズム17）と，3倍を求めて2倍するもの（アルゴリズム18）です。

7)　7倍

7倍にも2通りの方法があります。8倍をして自分自身を減算するもの（アルゴリズム19），6倍をして自分自身を加算するもの（アルゴリズム20）です。

8)　その他

リスト3のプログラムにはこのほかにも，9倍，11倍，13倍，15倍，17倍についての例を示してありますが，残念ながらこの倍率では，ここで使用しているαCやTurbo CやXCは参考になりません。近くに，MS-CとかLSI-C，あるいはSun-3のUNIX上のCコンパイラを利用できる環境があれば，出力されるコードを覗いてみてください。加減算とシフトをほとんどパズルを解くように組み合わせて掛け算を行っているのがわかるでしょう。

そこまで頑張っていなくても，それほど大きな倍率でない掛け算（1倍，2倍，3倍，4倍程度）の場合は，これまで見てきたように，多くのコンパイラで実際の掛け算（命令やサブルーチンによる）を避けるために，なんらかの工夫がされています。掛け算命令を備えているCPUのコンパイラにはこのような努力はあまりされていないのが現状ですが，そのようなコンパイラは努力不足を非難されても仕方がないでしょう。

しかし，掛け算をやみくもに加減算やシフトに置き換えるのが必ずしもいい結果を生み出すものではないということも認識しておかなければなりません。たとえば，1000倍をする場合，自分自身を10ビット左シフト（1024倍）したものから，4ビット左シフトをしたもの（16倍）と3ビット左シフト（8倍）をしたものの和（24倍）を引くことで答えを得ることができます。

もっとも，これらの命令の実行時間の合

リスト 3

1. 乗算(C言語)

```
int  i,j;
main()
{
        j = 1 * i;
        j = 2 * i;
        j = 3 * i;
        j = 4 * i;
        j = 5 * i;
        j = 6 * i;
        j = 7 * i;
        j = 9 * i;
        j = 11 * i;
        j = 13 * i;
        j = 15 * i;
        j = 17 * i;
}
```

2. 乗算(αC)

```
;
;       このリストはαCのオブジェクトを
;       逆アセンブルして作成したものです
;
;
;       変数のベースアドレスを1000Hとした場合
;
_i      EQU     1000H
_j      EQU     1002H
;
SMUL    EQU     023FH           ; 符号付き乗算
;
_main:
        PUSH    BC
;
        LD      HL,(_i)
        LD      (_j),HL         ; 1*i=i
;
        LD      HL,(_i)
        ADD     HL,HL           ;  2*i=i+i
        LD      (_j),HL
;
        LD      HL,(_i)
        LD      DE,3
        CALL    SMUL            ; 3*i
        LD      (_j),HL
;
        LD      HL,(_i)
        ADD     HL,HL           ; 2*i
        ADD     HL,HL           ; 4*i
        LD      (_j),HL
;
        LD      HL,(_i)
        LD      DE,5
        CALL    SMUL            ; 5*i
        LD      (_j),HL
;
        LD      HL,(_i)
        LD      DE,6
        CALL    SMUL            ; 6*i
        LD      (_j),HL
;
        LD      HL,(_i)
        LD      DE,7
        CALL    SMUL            ; 7*i
        LD      (_j),HL
;
        LD      HL,(_i)
        LD      DE,9
        CALL    SMUL            ; 9*i
        LD      (_j),HL
;
        LD      HL,(_i)
        LD      DE,11
        CALL    SMUL            ; 11*i
        LD      (_j),HL
;
        LD      HL,(_i)
        LD      DE,13
        CALL    SMUL            ; 13*i
        LD      (_j),HL
;
        LD      HL,(_i)
        LD      DE,15
        CALL    SMUL            ; 15*i
        LD      (_j),HL
;
        LD      HL,(_i)
        LD      DE,17
        CALL    SMUL            ; 17*i
        LD      (_j),HL
;
        POP     BC
        RET
```

3. 乗算(Turbo C)

```
;
;       このリストは Turbo C の出力リスト
;       にコメントを付け加えたものです
;
        name    mul
_TEXT   segment byte public 'CODE'
DGROUP  group   _BSS
        assume  cs:_TEXT,ds:DGROUP,ss:DGROUP
_TEXT   ends
_DATA   segment word public 'DATA'
_d@     label   byte
_DATA   ends
_BSS    segment word public 'BSS'
_b@     label   byte
_BSS    ends
_TEXT   segment byte public 'CODE'
; Line 3
_main   proc    near
; Line 4
        mov     ax,word ptr DGROUP:_i           ; 1*i
        mov     word ptr DGROUP:_j,ax
; Line 5
        mov     ax,word ptr DGROUP:_i
        shl     ax,1                            ; 2*i
        mov     word ptr DGROUP:_j,ax
; Line 6
        mov     ax,word ptr DGROUP:_i
        mov     dx,3
        mul     dx                              ; 3*i
        mov     word ptr DGROUP:_j,ax
; Line 7
        mov     ax,word ptr DGROUP:_i
        shl     ax,1                            ; 2*i
        shl     ax,1                            ; 4*i
        mov     word ptr DGROUP:_j,ax
; Line 8
        mov     ax,word ptr DGROUP:_i
        mov     dx,5
        mul     dx                              ; 5*i
        mov     word ptr DGROUP:_j,ax
; Line 9
        mov     ax,word ptr DGROUP:_i
        mov     dx,6
        mul     dx                              ; 6*i
        mov     word ptr DGROUP:_j,ax
; Line 10
        mov     ax,word ptr DGROUP:_i
        mov     dx,7                            ; 7*i
        mul     dx
        mov     word ptr DGROUP:_j,ax
; Line 11
        mov     ax,word ptr DGROUP:_i
        mov     dx,9                            ; 9*i
        mul     dx
        mov     word ptr DGROUP:_j,ax
; Line 12
        mov     ax,word ptr DGROUP:_i
        mov     dx,11
        mul     dx                              ; 11*i
        mov     word ptr DGROUP:_j,ax
; Line 13
        mov     ax,word ptr DGROUP:_i
        mov     dx,13
        mul     dx                              ; 13*i
        mov     word ptr DGROUP:_j,ax
; Line 14
        mov     ax,word ptr DGROUP:_i
        mov     dx,15
        mul     dx                              ; 15*i
        mov     word ptr DGROUP:_j,ax
; Line 15
        mov     ax,word ptr DGROUP:_i
        mov     dx,17                           ; 17*i
        mul     dx
        mov     word ptr DGROUP:_j,ax
; Line 16
        .ret
_main   endp
_TEXT   ends
_BSS    segment word public 'BSS'
        public  _i
_i      label   word
        db      2 dup (?)
        public  _j
_j      label   word
        db      2 dup (?)
_BSS    ends
_DATA   segment word public 'DATA'
_s@     label   byte
_DATA   ends
_TEXT   segment byte public 'CODE'
        public  _main
_TEXT   ends
        end
```

4. 乗算(XC)

```
;
;       このリストは XC の出力リストに
;       コメントを付け加えたものです
;
        include fefunc.h
        .COMM   _i,4
        .COMM   _j,4
        .XREF   __main
        .XDEF   _main
        .TEXT
_main:
__main:
        BRA     L1
L2:
        MOVE.L  _i,_j               ; 1*i
;
        MOVE.L  _i,D0
        ASL.L   #1,D0               ; 2*i
        MOVE.L  D0,_j
        MOVE.L  _i,D0
;
        MOVE.L  #3,D1
        MOVEM.L D0/D1,-(SP)
        FPACK   __CLMUL             ; 3*i
        MOVEM.L (SP)+,D0/D1
        MOVE.L  D0,_j
;
        MOVE.L  _i,D0
        ASL.L   #2,D0               ; 4*i
        MOVE.L  D0,_j
;
        MOVE.L  _i,D0
        MOVE.L  #5,D1
        MOVEM.L D0/D1,-(SP)
        FPACK   __CLMUL             ; 5*i
        MOVEM.L (SP)+,D0/D1
        MOVE.L  D0,_j
;
        MOVE.L  _i,D0
        MOVE.L  #6,D1
        MOVEM.L D0/D1,-(SP)
        FPACK   __CLMUL             ; 6*i
        MOVEM.L (SP)+,D0/D1
        MOVE.L  D0,_j
;
        MOVE.L  _i,D0
        MOVE.L  #7,D1
        MOVEM.L D0/D1,-(SP)
        FPACK   __CLMUL             ; 7*i
        MOVEM.L (SP)+,D0/D1
        MOVE.L  D0,_j
;
        MOVE.L  _i,D0
        MOVE.L  #9,D1
        MOVEM.L D0/D1,-(SP)
        FPACK   __CLMUL             ; 9*i
        MOVEM.L (SP)+,D0/D1
        MOVE.L  D0,_j
;
        MOVE.L  _i,D0
        MOVE.L  #11,D1
        MOVEM.L D0/D1,-(SP)
        FPACK   __CLMUL             ; 11*i
        MOVEM.L (SP)+,D0/D1
        MOVE.L  D0,_j
;
        MOVE.L  _i,D0
        MOVE.L  #13,D1
        MOVEM.L D0/D1,-(SP)
        FPACK   __CLMUL             ; 13*i
        MOVEM.L (SP)+,D0/D1
        MOVE.L  D0,_j
;
        MOVE.L  _i,D0
        MOVE.L  #15,D1
        MOVEM.L D0/D1,-(SP)
        FPACK   __CLMUL             ; 15*i
        MOVEM.L (SP)+,D0/D1
        MOVE.L  D0,_j
;
        MOVE.L  _i,D0
        MOVE.L  #17,D1
        MOVEM.L D0/D1,-(SP)
        FPACK   __CLMUL             ; 17*i
        MOVEM.L (SP)+,D0/D1
        MOVE.L  D0,_j
L3:
        UNLK    A6
        RTS
L1:
        LINK    A6,#0
        BRA     L2
        .DATA
        .EVEN
        .END
```

計が掛け算命令で1000倍をする場合より速くなるかどうかはわかりません。特にメモリのウエイト数が多いときは，命令のフェッチが遅くなる分，命令数が多い加減算とシフトの組み合わせは不利になります。このため，どこら辺で見切りをつけて実際に掛け算（命令やサブルーチン）を行ったらよいかはかなり難しい問題なのです。ちなみにSun-3のUNIX上のCコンパイラは，定数値を掛ける場合は絶対に加減算とシフトの組み合わせを用いて計算しています(少なくとも1000倍まではそうでした)。

配列要素へのアクセス

高級言語において配列というデータ構造は頻繁に用いられます。当然，アセンブリ言語でプログラミングを行う場合も配列の使用頻度は高いと思われます。アセンブリ言語で配列を実現する場合，１次元配列なら何とかなりそうですが，２次元配列，３次元配列，４次元配列，……，はどうすればよいのでしょうか。といっても，現実の使用は２次元までの配列でこと足りることが多い（αCは２次元配列までしかサポートしていません）ため，ここでは１次元配列と２次元配列の要素へアクセスするためのコードを見ることにしましょう。

リスト４がαC, Turbo C, XCの各コンパイラが１次元配列，２次元配列の要素をアクセスする場合のコードですが，これらはまったく同一のコード（アルゴリズム）みたいです。コンパイラのコードの出し方がひとつしかないということは，その方法が配列要素をアクセスするための常識ということなのでしょう。

１次元配列の要素にアクセスするための方法をアルゴリズム21に，２次元配列の要素にアクセスするための方法をアルゴリズム22に示します。なお，アルゴリズム21や22はやたら複雑そうですが，添字の値が定数で与えられた場合は，コンパイルをする時点で要素のサイズとの掛け算ができるので，もっと少ない手順になります。たとえば，int型の配列を読む，

```
v=a[2]
```

という式に対するコードを，ばか正直に，

```
move.l   #2 , d0
asl.l    #2 , d0
add.l    #_a , d0
move.l   d0 , a0
move.l   (a0) , _v
```

とする（68000のコードの場合）タコなコンパイラはまずないはずで，通常は，

```
move.l   _a+8 , _v
```

というコードを出すでしょう。この場合の，

```
_a + 8
```

の８という値は，

２(添字の値)×４(int型のサイズ)

という式をコンパイラが計算したものなのです。

アルゴリズム21と22についてもっと解説しましょう。C言語において，１次元配列はメモリ内に添字が小さい順に割り当てられます。メモリのアドレス空間は１次元ですから，これは自然な割り当てといえます。したがって，ある添字を与えたときの配列要素は，配列の先頭アドレスから，

添字×配列要素のサイズ（バイト）

離れたアドレスでアクセスできるのです。これがアルゴリズム21の意味です（説明するほどのことでもないか)。

```
アルゴリズム10  j=1*i
  j ← i
アルゴリズム11  j=2*i
  j ← iを1ﾋﾞｯﾄ左ｼﾌﾄ
アルゴリズム12  j=2*i
  j ← i + i
アルゴリズム13  j=3*i
  tmp ← i * 2
          (ｱﾙｺﾞﾘｽﾞﾑ11または12)
  j ← tmp + i
アルゴリズム14  j=4*i
  j ← iを2ﾋﾞｯﾄ左ｼﾌﾄ
アルゴリズム15  j=4*i
  tmp ← i + i
  j ← tmp + tmp
アルゴリズム16  j=5*i
  tmp ← i * 4
          (ｱﾙｺﾞﾘｽﾞﾑ14または15)
  j ← tmp + i

アルゴリズム17  j=6*i
  tmp ← i + i
  tmp2 ← tmp + tmp
  j ← tmp + tmp2
アルゴリズム18  j=6*i
  tmp ← i * 3
          (ｱﾙｺﾞﾘｽﾞﾑ13)
  j ← tmp * 2
          (ｱﾙｺﾞﾘｽﾞﾑ11または12)
アルゴリズム19  j=7*i
  tmp ← iを3ﾋﾞｯﾄ左ｼﾌﾄ(8倍)
  j ← tmp - i
アルゴリズム20  j=7*i
  tmp ← i * 6
          (ｱﾙｺﾞﾘｽﾞﾑ17または18)
  j ← tmp + i
```

一方，２次元配列をメモリに割り当てる場合は，２次元のデータ構造を１次元のメモリに適合するように並べ換えなくてはなりません。C言語では２次元配列をメモリに割り当てる場合は，行の番号の小さいものが先（低いアドレス）に，同じ行番号では列の番号の小さいものが先になるようにしています(ちなみにFORTRANの２次元配列は，列の番号の小さいものが先に，同じ列番号では行の番号の小さいものが先になっている)。たとえば，int型の２×３の２次元配列，

```
int  A[2][3] ;
```

では，それぞれの要素は，メモリ上に，

A[0][0]	第０行	第０列
A[0][1]	第０行	第１列
A[0][2]	第０行	第２列
A[1][0]	第１行	第０列
A[1][1]	第１行	第１列
A[1][2]	第１行	第２列
A[2][0]	第２行	第０列
A[2][1]	第２行	第１列
A[2][2]	第２行	第２列

という順序で割り当てられます。

この順序を見れば，行の番号がｉ，列の番号がｊであるような２次元配列の要素をアクセスするためには，まず最初は行の番号に注目してA[i][0]の位置（アドレス）を求め，そこからは１次元配列でｊ番目の要素をアクセスするのと同様にすればよいことがわかります。これがアルゴリズム22の意味です。ある行番号，たとえばｉに対する第０列のアドレスは，その直前までの要素の個数を計算すれば求まります。またアルゴリズム21と22は配列要素を読む場合を示していますが，配列に書き込む場合も同様です。

ところで，配列をアクセスするアルゴリズム自身は１種類しかありませんが，それを実際に行うときのコードの出し方（どういう命令やアドレッシングを用いるか）にはいろいろな方法があります。この配列へのアクセスの方法がどのような命令列で実現されているのかを見れば，コンパイラ設計者のセンスが判ってしまうくらいのバリエーションがあります。もちろん，Z80などのアドレッシングモードの種類が少ないCPUのコンパイラにおいては，誰が考えても同じような命令列になってしまいますが，68000などの豊富なアドレッシングモードを持つCPUのコンパイラでは読む人をうならせるようなコードを出してほしいものですね。

リスト４で，XCは，

```
    v=b[3][j]
```
という記述に対して,

```
    move.l  _j , d0
    asl.l   #2 , d0
    add.l   #120+_b , d0
    move.l  d0 , a0
    move.l  (a0) , _v
```
という無難なコードを出していますが，Sun-3のUNIX上のCコンパイラ（68010コードを出力させたとき）では,

```
    move.l  _j , d0
    asl.l   #2 , d0
    lea     _b+120 , a0
    move.l  0(a0 , d0.l) , _v
```
というコードを出力して，インデックスアドレッシングはこのように使うのだという例を見せてくれるのです(もちろん68000でも実行可能なコードですよ)。

最後に

ここまで，いくつかのC言語の記述について，Cコンパイラがどのようなコードを出力するかを見てきました。今回はページ数の都合などから，ループ，選択，掛け算，配列に関するC言語の記述に対するコードしか紹介することができませんでしたが，そのなかにもなるほどとうならせるようなコードがあったと思います。これ以外の例としては,

構造体のアクセス
共用体のアクセス
ビットフィールド

の記述に対するコードも面白そうなのですが，これらについては短いサンプルを作って，自分でコンパイルしてみることを勧めます。

アルゴリズム21　v=A[i]

```
tmp ← i
tmp ← tmp * 要素のｻｲｽﾞ
tmp2 ← a[0][0]のｱﾄﾞﾚｽ
v ← *(tmp2 + tmp)
```

注）　*(tmp2+tmp)はtmp2+tmpで計算されるアドレスで示されるメモリの内容を示す。

アルゴリズム22　v=B[i][j]

```
tmp ← i
tmp ← tmp *  要素のｻｲｽﾞ
tmp ← tmp *  宣言した行の数
              (添字の最大値+1)
tmp2 ← j
tmp2 ← tmp2 * 要素のｻｲｽﾞ
tmp3 ← b[0][0]のｱﾄﾞﾚｽ
v  ← *(tmp3 + tmp2 + tmp)
```

注）　*(tmp3+tmp2+tmp)はtmp3+tmp2+tmpで計算されるアドレスで示されるメモリの内容を示す。

たいていのCコンパイラにはアセンブリ言語の出力を生成するオプションが備えられています（残念ながらαCにはないのですが）から，それを使えば，出てきたオブジェクトプログラムをデバッガで追うという煩わしい操作をしなくてすむでしょう。

そして，Cコンパイラの出力するコードを見る場合はできるだけ多くのコンパイラのコードを比較するのがいいと思います。実は今回，9種類のCコンパイラのコードを比べてみた（もろもろの事情ですべてを紹介できませんが）のですが，それぞれの出力が少しずつ違っていて，同じことをするのにこんなにもやり方があるのかと大変参考になりました。

また，賢そうに見える（最適化を売り物にしてる）コンパイラでも，ある場合ではどうしようもないコードを出していたり，またその逆もあったりで，なかなか楽しい時を過ごせたのです（うーん，少し異常かな)。

さて，来月号では今回得た知識をベースに，C言語とアセンブリ言語とをリンクして活用する特別講義を，もう一度開講する予定です。では，今日のところはこのくらいでおしまいにしましょう。

リスト 4

1.　配列(C言語)

```
int     a[10],b[10][10];
int     i,j,v;
main()
{
        v = a[3];
        v = a[i];
        v = b[3][4];
        v = b[i][4];
        v = b[3][j];
        v = b[i][j];
}
```

2.　配列(αC)

```
;
;       このリストはαCのオブジェクトを
;       逆アセンブルして作成したものです
;
;       変数のベースアドレスを1000Hとした場合
;
_a      EQU     1000H
_b      EQU     1014H
_i      EQU     10DCH
_j      EQU     10DEH
_v      EQU     10E0H
;
USMUL   EQU     26BH            ; 符号なし乗算

_main:
        PUSH    BC
;
        LD      HL,(_a+6)       ; a[3]
        LD      (_v),HL
;
        LD      HL,(_i)
        ADD     HL,HL           ; 2*i
        LD      DE,_a
        ADD     HL,DE           ; a[i]
        LD      A,(HL)
        INC     HL
        LD      H,(HL)
        LD      L,A
        LD      (_v),HL
;
        LD      HL,(_b+68)      ; 68=3*10*2+4*2 b[3][4]
        LD      (_v),HL
;
        LD      HL,(_i)
        LD      DE,20
        CALL    USMUL           ; 20*i=10*i*2
        LD      DE,_b+8
        ADD     HL,DE           ; b[i][4]
        LD      A,(HL)
        INC     HL
        LD      H,(HL)
        LD      L,A
        LD      (_v),HL
;
        LD      HL,(_j)
        ADD     HL,HL           ; 2*j
        LD      DE,_b+60        ; 60=3*10*2      b[3][0]
        ADD     HL,DE           ; b[3][j]
        LD      A,(HL)
        INC     HL
        LD      H,(HL)
        LD      L,A
        LD      (_v),HL
;
        LD      HL,(_i)
        LD      DE,20
        CALL    USMUL           ; 20*i
        LD      DE,_b
        ADD     HL,DE           ; b[i][0]
        PUSH    HL
        LD      HL,(_j)
        ADD     HL,HL           ; 2*j
        POP     DE
        ADD     HL,DE           ; b[i][j]
        LD      A,(HL)
        INC     HL
        LD      H,(HL)
        LD      L,A
        LD      (_v),HL
;
        POP     BC
        RET
```

3.　配列(Turbo C)

```
;
;       このリストは Turbo C の出力リスト
;       にコメントを付け加えたものです
;
        name    ind
_TEXT   segment byte public 'CODE'
DGROUP  group   _BSS
        assume  cs:_TEXT,ds:DGROUP,ss:DGROUP
_TEXT   ends
_DATA   segment word public 'DATA'
_d@     label   byte
_DATA   ends
_BSS    segment word public 'BSS'
_b@     label   byte
_BSS    ends
_TEXT   segment byte public 'CODE'
; Line 4
_main   proc    near
; Line 5
        mov     ax,word ptr DGROUP:_a+6         ; a[3]
        mov     word ptr DGROUP:_v,ax
; Line 6
        mov     bx,word ptr DGROUP:_i
        shl     bx,1                            ; 2*i
        mov     ax,word ptr DGROUP:_a[bx]       ; a[i]
        mov     word ptr DGROUP:_v,ax
; Line 7
        mov     ax,word ptr DGROUP:_b+68        ; b[3][4]
        mov     word ptr DGROUP:_v,ax
; Line 8
        mov     ax,word ptr DGROUP:_i
        mov     dx,20
        mul     dx                              ; 20*i
        mov     bx,ax
        mov     ax,word ptr DGROUP:_b[bx+8]     ; b[i][4]
        mov     word ptr DGROUP:_v,ax
; Line 9
        mov     bx,word ptr DGROUP:_j
        shl     bx,1                            ; 2*j
        mov     ax,word ptr DGROUP:_b[bx+60]    ; b[3][j]
        mov     word ptr DGROUP:_v,ax
; Line 10
        mov     ax,word ptr DGROUP:_i
        mov     dx,20
        mul     dx                              ; 20*i
        mov     bx,ax
        mov     ax,word ptr DGROUP:_j
        shl     ax,1                            ; 2*j
        add     bx,ax
        mov     ax,word ptr DGROUP:_b[bx]       ; b[i][j]
        mov     word ptr DGROUP:_v,ax
; Line 11
        ret
_main   endp
_TEXT   ends
_BSS    segment word public 'BSS'
        public  _v
_v      label   word
        db      2 dup (?)
        public  _b
_b      label   word
        db      200 dup (?)
        public  _i
_i      label   word
        db      2 dup (?)
        public  _a
_a      label   word
        db      20 dup (?)
        public  _j
_j      label   word
        db      2 dup (?)
_BSS    ends
_DATA   segment word public 'DATA'
_s@     label   byte
_DATA   ends
_TEXT   segment byte public 'CODE'
        public  _main
_TEXT   ends
        end
```

4.　配列(XC)

```
;
;       このリストは XC の出力リストに
;       コメントを付け加えたものです
;
        include fefunc.h
        .COMM   _a,40
        .COMM   _b,400
        .COMM   _i,4
        .COMM   _j,4
        .COMM   _v,4
        .XREF   __main
        .XDEF   _main
        .TEXT
_main:
__main:
        BRA     L1                      ; size=4
L2:
        MOVE.L  12+_a,_v                ; a[3]   12=3*size
;
        MOVE.L  _i,D0
        ASL.L   #2,D0                   ; i*size
        ADD.L   #_a,D0
        MOVE.L  D0,A0                   ; a[i] のアドレス
        MOVE.L  (A0),_v
;
        MOVE.L  136+_b,_v               ; b[3][4] 136=3*size*10+4*size
;
        MOVE.L  _i,D0
        MOVE.L  #40,D1
        MOVEM.L D0/D1,-(SP)
        FPACK   __CLMUL                 ; 40*i  40=10*size
        MOVEM.L (SP)+,D0/D1
        ADD.L   #16+_b,D0               ; b[i][4] のアドレス
        MOVE.L  D0,A0                   ; 16=4*size
        MOVE.L  (A0),_v
;
        MOVE.L  _j,D0
        ASL.L   #2,D0                   ; j*size
        ADD.L   #120+_b,D0              ; b[3][j] のアドレス
        MOVE.L  D0,A0                   ; 120=3*size*10
        MOVE.L  (A0),_v
;
        MOVE.L  _i,D0
        MOVE.L  #10,D1
        MOVEM.L D0/D1,-(SP)
        FPACK   __CLMUL                 ; 10*i
        MOVEM.L (SP)+,D0/D1
        ADD.L   _j,D0                   ; 10*i+j
        ASL.L   #2,D0                   ; size*(10*i+j)
        ADD.L   #_b,D0                  ; b[i][j] のアドレス
        MOVE.L  D0,A0
        MOVE.L  (A0),_v
L3:
        UNLK    A6
        RTS
L1:
        LINK    A6,#0
        BRA     L2
        .DATA
        .EVEN
        .END
```

Appendix C言語簡易リファレンス

C言語には覚えなければいけない書式や文法に関する約束ごとはそれほど多くはありません。逆にいえば文法よりも，プログラムの動作に関する本質的な理解が重要ということです。プログラミング時や，掲載されたプログラムテキストを読む場合，必要に応じてこのリファレンスを参照するとよいでしょう。

このリファレンスはXCの書式に基づいて作成されたもので，ランゲージシリーズ（αC）やSmall Cといったサブセット版のCコンパイラではサポートされていない機能もあるのでご注意ください。

関数

プログラムは関数の集合として表現されます。各関数の処理は｛と｝でくくられた内容を実行したあと呼び出されたところに復帰します。また，すべての関数は個別にコンパイルすることができますが，プログラムには最初に実行されるmain関数がどこかにあることが必要です。

関数の宣言（フォワード宣言）

　関数の型　関数名（引数の型）;

　　（引数の型宣言は省略可。宣言したものを，特にプロトタイプ宣言ということもある）

関数の定義

```
関数の型　関数名（引数リスト）
引数宣言
｛
    変数の宣言
    関数の本体
｝
```

関数main

　プログラムは必ずこのmain( )から実行される。

```
void  main(argc, argv, envp)
    int      argc;
    char     * argv[ ];
    char     * envp[ ];
```

　引数の意味は

　　argc……コマンドラインパラメータの個数

　　argv……各コマンドラインパラメータを指すポインタの配列

　　envp……オペレーティングシステムの環境変数を指すポインタの配列

文

変数の宣言や関数呼び出し，代入文などの式には必ず；を付ける必要があります。また，Cには行の概念はなく，；によって複数の文を列記することも可能です。

空文　；

単文　式；

複文　｛文　文 … 文｝

制御構造

Cにはプログラムの流れを管理するものとして，次のような制御文が用意されています。同時にいくつもの文を制御したい場合には複文を用いることができ，空ループなどを作りたい場合には空文を使用します。

while(式) 文

　式の値が真である間，文を繰り返し実行する。

do 文 while (式);

　文を実行し，式の値が真であれば繰り返す。

for (初期化式；条件式；ループ文) 文

　初期化式を実行し，そのあと条件式，文，ループ文の順に，条件式の値が真である間繰り返し評価・実行する。

break;

　ループを強制的に終わらせる。

continue;

　次の繰り返しの最初に実行を移す。

if (式) 文1　else 文2

　式の値が真なら文1を，偽なら文2を実行する。

```
switch (式) ｛
    case 定数式：文
    case 定数式：文
      ：
    default：   文
｝
```

　式の値が定数式と一致するcase以降（どの定数式とも一致しなければ，default以降）の文をすべて実行する。ふつうは文の最後にbreak文を入れて，実行が終わるとブロックの外に脱出できるようにする。

return (式);

return;

　関数の戻り値を式の値として関数を終了させる。2番目の書式はvoid型関数のもの。

goto ラベル

　無条件にラベルの場所へジャンプする。ラベルは同じ関数の中になくてはならない。

データ型

Cでは豊富なデータ型をサポートしていますが，最もよく使われるのはchar型とint型で，これにdouble型くらいを覚えておけばよいでしょう。

整数型

符号付き	
char（文字型）	8ビット
int	32(16)ビット
short int (short)	16ビット
long int (long)	32ビット
符号なし（正整数）	
unsigned char（文字型）	8ビット
unsigned int (unsigned)	32ビット
unsigned short int (unsigned short)	16ビット
unsigned long int (unsigned long)	32ビット

浮動小数点型

float	32ビット
double	64ビット

その他

　enum（列挙型）

　void（戻り値がない関数の型）

注）上記のビット長はXCに準拠したものです。なお，8ビット機または8086用のCの場合，int型は16ビットとなります。

定数表現

Cで扱う定数には次のようなものがあり，それぞれに特有の表記方法がとられています。

整数定数（int型になるが，定数の終わりにLか1を加えるとlong型になる）

　10進数　nnn（0で始まってはいけない）nは数字。

　8進数　0nnn

　16進数　0xnnn

浮動小数点定数（double型になる）

小数点表現　nn. nnn

指数表現　nn. nnE±nn

文字定数（char型になる）

'１バイトの文字'　文字にはエスケープシーケンスも含まれる。

文字列定数（char型の配列になる）

"文字列"　エスケープシーケンスを含んでよい。

エスケープシーケンス……特殊な文字の表現に使う。１バイト。

¥n	復帰改行
¥t	水平タブ
¥v	垂直タブ
¥b	バックスペース
¥r	復帰
¥f	改ページ
¥a	警告ベル
¥'	シングルクォーテーション（'）
¥"	ダブルクォーテーション（"）
¥¥	円記号（¥）
¥nnn	８進数で表されたASCII文字
¥xnnn(¥Xnnn)	16進数で表されたASCII文字

変数の宣言　変数は使用する前に前記のデータ型を宣言しなければなりません。ただし先頭の記憶クラスは，多くの場合省略することができます。

enum型以外

記憶クラス　型名　変数名＝初期化定数，変数名，…，変数名；

enum型

enum　タグ名｛識別子リスト｝変数名；

変数名を省略すると，型宣言になる。

記憶クラス　Cでは変数の記憶方法によって，関数が呼び出されたときにだけメモリ上のスタック領域に変数が発生する動的な変数と，メモリに常駐する静的な変数，そしてレジスタに直接記憶するレジスタ変数があります。また，変数が通用する範囲も大域的なものと局所的なものとに分かれます。これらの概念をまとめたものが次の４つの記憶クラスです。

auto	内部レベル	メモリ上，動的	宣言ブロック内のみ
register	内部レベル	レジスタ内	宣言ブロック内のみ
static	任意のレベル	メモリ上，静的	宣言ブロック内のみ
extern	任意のレベル	メモリ上，静的	プログラムのすべての範囲
typedef	型名の定義		

配　列　同じデータ型の要素の集合です。配列の添え字には[ ]が利用されます。BASIC と違ってA[10]と宣言すると０～９の10個が確保されます。

配列の宣言

配列の型　配列名[要素数][要素数]…[要素数]＝｛初期化要素｝；

要素数と初期化要素は定数式でなくてはならない。なお，初期化要素は省略可。

構造体　型の異なる複数の変数や配列を組み合わせた集合を，ひとつの新しいデータ単位として扱うというものです。

構造体の宣言

struct　タグ名｛メンバ名；…；メンバ名；｝変数名＝｛初期化要素｝；

タグ名は構造体の型名。

タグ名，変数名，初期化要素は省略可。

ビットフィールドでメンバを宣言できる。書式は次のとおり。

unsigned メンバ名：定数式；

構造体の参照

構造体変数名.メンバ名

構造体へのポインタ変数名－＞メンバ名

共用体　同じ領域に格納された値を，複数の異なる変数名で共用できるようにするものです。それぞれの変数は違う型であってもかまいません。

共用体の宣言

union　タグ名｛メンバ宣言リスト｝変数名＝｛初期化要素｝；

タグ名は構造体の型名。

タグ名，変数名，初期化要素は省略可。

ビットフィールドの指定は不可。

共用体の参照

共用体変数名.メンバ名

共用体へのポインタ変数名－＞メンバ名

演算子　Cには非常に多くの演算子が用意されています。これらのすべてを覚えておく必要はありませんが，いずれもプログラミングの実情に合わせて作られたものです。特にインクリメント・デクリメント演算子は，プログラマにとって利用効果の高いものでしょう。

算術演算子（演算の結果を式の値とする）

a＋b	加算（a足すb）を行う。
a－b	減算（a引くb）を行う。
－a	負符号をつける（マイナスa）。
a＊b	乗算（a掛けるb）を行う。
a／b	除算（a割るb）を行う。
a％b	剰余（aをbで割った余り）をとる。

関係演算子（式の値は，真のとき１で，偽のとき０）

a＝＝b	aとbは等しい。
a！＝b	aとbは等しくない。
a＜b	aはbより小さい。
a＜＝b	aはb以下である。
a＞b	aはbより大きい。
a＞＝b	aはb以上である。

論理演算子（式の値は関係演算子と同じ。０以外は真とみなす）

a&&b	論理積（AND：aが真かつbも真）をとる。
a ‖ b	論理和（OR：aが真またはbが真）をとる。
！a	論理否定（NOT：aが偽）をとる。

インクリメント・デクリメント演算子

a＋＋	aを式の値とし，a＋１の値をaに代入する。
＋＋a	a＋１の値をaに代入し，それを式の値とする。
a－－	aを式の値とし，a－１の値をaに代入する。
－－a	a－１の値をaに代入し，それを式の値とする。

ビット演算子（結果を式の値とする）

a＆b	ビットごとの論理積（AND）をとる。
a｜b	ビットごとの論理和（OR）をとる。
a＾b	ビットごとの排他的論理和（XOR）をとる。
～a	ビットごとに反転する（NOT）。

シフト演算子

a＞＞b	aの値をbビット右にシフトし，式の値とする。
a＜＜b	aの値をbビット左にシフトし，式の値とする。

代入演算子（Cでは代入も式として扱う。代入した値(右辺値)を式の値とする）

a＝b	bの値をaに代入する。
a＋＝b	a＋bの値をaに代入する。
a－＝b	a－bの値をaに代入する。
a＊＝b	a＊bの値をaに代入する。
a／＝b	a／bの値をaに代入する。
a％＝b	a％bの値をaに代入する。
a＆＝b	a＆bの値をaに代入する。

a｜=b　　a｜bの値をaに代入する。
a^=b　　a^bの値をaに代入する。
a>>=b　　a>>bの値をaに代入する。
a<<=b　　a<<bの値をaに代入する。

条件演算子

a？b：c　　aを評価し，真ならb，偽ならcを式の値とする。

カンマ演算子（逐次評価）

a，b，…，c　　式を左から順に評価し，最後に評価した値を式の値とする。

アドレス演算子と間接演算子

&a　　aのアドレスを式の値とする。
＊a　　ポインタ変数aが指すアドレスの内容を式の値とする。

メンバ参照演算子（参照した値を式の値とする）

a.b　　構造体・共用体aのメンバbを参照する。
a−>b　　構造体・共用体aへのポインタからメンバbを参照する。

sizeof 演算子

sizeof a　　aに割り当てられている記憶領域のバイト数を値とする。

型キャスト演算子

(型名)a　　aの値を（型名）の型に変換し，式の値とする。

プリプロセッサ

Cではコンパイルに先立って，さまざまなマクロ定義を展開したり，外部ファイルを取り込んだりする前処理が行われます。このプリプロセッサ機能を生かすことにより，ソースリストを簡潔な読みやすいものにできるわけです。

マクロ定義

＃define　識別子　文字列
＃define　識別子（引数リスト）　文字列
ソースプログラム中の識別子を文字列で置き換える。
文字列は省略可だが，その場合識別子は削除される。
＃undef　識別子
＃defineによるマクロ定義を取り消す。
＃define～＃undefで，マクロ定義の範囲を指定できる。

ファイル取り込み

＃include　“ファイル名”
＃include　〈ファイル名〉
ファイル名で指定されたインクルードファイルを取り込む。

条件付きコンパイル

＃if　　コンパイル抑止条件式
⋮　　　　ブロック；
＃elif　　コンパイル抑止条件式
⋮　　　　ブロック；
＃elif　　コンパイル抑止条件式
⋮　　　　ブロック；
＃endif
コンパイル抑止条件式は定数式。
この値が真（0以外）であるブロックをコンパイルする。
defined　識別子
コンパイル抑止条件式として使う。
識別子がすでに＃defineで定義されていれば真になる。
＃ifdef　識別子
＃ifndef　識別子
＃ifdefは＃if definedと同じ。＃ifndefはその逆。

行制御

＃line　定数　“ファイル名”
コンパイラ内部に記憶されている行番号とファイル名を変更する。

代表的な入出力関数

よく使用される基本的な入出力関数についてまとめておきましょう。入出力対象は特に指定しなければコンソールになります。つまり，キーボードが標準入力でスクリーンが標準出力になるわけです。ただし，入出力リダイレクションにより，ファイル等に変更することも可能です。

1文字入出力

プリプロセッサで“conio.h”をインクルードしておく。
putch(c)；
引数：int
戻り値：void（なし）
機能：コンソールに文字cを出力する。
getch()；
getche()；
引数：なし
戻り値：int
機能：コンソールから1文字読み込み，その値を返す。
getcheには入力エコーバックあり。

文字列入出力

プリプロセッサで“stdio.h”をインクルードしておく。
printf(format_string[, arg, …])；
引数：(format_string)　書式文字列へのポインタ（ふつうは文字列を直接書く）
(arg)　任意の型
戻り値：int（出力した文字数を返す）
機能：連続した文字を標準出力stdoutに出力する。
引数argがあれば，format_stringに指定した書式で出力する。
表示文字列と，argの書式文字列を混在させることができる。
書式はさまざまに指定できるが，主な例を以下に示す。
printf(“変数の値はそれぞれ%d，%dです”，dec1，dec2)；
符号付き10進整数を出力する。
printf(“%.5u個のファイルが存在します”，dec)；
符号なし10進整数を5桁の欄に出力する。
printf(“アドレス：%.4X”，hex)；
符号なし16進整数を4桁の欄に出力する。
printf(“%#.10fが解です”，dbl)；
double値を小数点以下10桁まで出力する。
printf(“あなたの体重は%eグラムです”，dbl)；
double値を指数表現で出力する。
printf(“押されたキーは%cです”，chr)；
1文字出力する。
printf(“氏名：%.8s”，str)；
文字列を最大8文字出力する。
このstrはポインタになっている。
scanf(format_string [, pointer, …])；
引数：(format_string)　書式文字列へのポインタ（ふつうは文字列を直接書く）
(pointer)　読み込みたい変数へのポインタ
戻り値：int（割り付けた領域の数を返す）
機能：標準入力stdinからデータを読み込み，format_string で指定された書式にしたがって変換し，pointer の示す領域に格納する（つまり変数の値として読み込む）。
書式指定の例を列挙する。
scanf(“ ”)；
空白文字を読み飛ばす。
空白文字は，空白' '・タブ'¥t'・改行'¥n'。
scanf(“%d %x”，&dec，&hex)；
10進，16進整数を読み込む。
scanf(“%e”，&flo)；
float値を読み込む。
scanf(“%E”，&dbl)；
double値を読み込む。
scanf(“%c”，&chr)；
1文字を読み込む。
scanf(“%8s”，str)；
文字列を最大8文字読み込む。
strはポインタだから，&をつけてはいけない。

C調言語講座 PRO-68K 第1回

まずはprintfより始めよ

この「C調言語講座PRO-68K」では，「Cの文法なんぞ知らなくても，まずはCを使うこと。そうすりゃなんとかなるものさ」なるいつもの名調子で，祝一平氏が手取り足取り４の字固め，ストロングスタイルでC言語術を毎日伝授してくれるそうです。さてさて，いったいどんな必殺技が飛び出すのでしょうか。

満開製作所 Iwai Ippei 祝 一平

今月からC compiler PRO-68K (以下XC) を対象としたC言語の講座を開くことになったわけである。

んが，最初に断っておくが私はCの文法に関してはそれほど詳しくはない。最近も，Cの文法に関していろいろ聞かれたことがあったのだが，いざ答えようとして，自分の無知さにあきれかえってしまった。それもそうなのである。実のところ，私がまともに読んだCの本といえば，『プログラミング言語C』(B.W.カーニハン，D.M.リッチー著，共立出版，2,500円)，通称「K&Rの教科書」だけなのである（そういえば，最初この本を読んだときは感動したっけ)。ちなみに，この本の日本語版は100刷を越えるという，とんでもないベストセラーになっている。私の持っているものは第8刷である。これはもしかすると，自慢になるのだろーか。

しかし，言わせてもらえれば，文法なんぞは，基本的にはうろ覚えでもいいのである。大事なのは，どういうプログラムを作るかなのである。英語の試験で満点を取ったとしても，その英語を使って何もしなければただの徒労なのである。文法なんぞはいくら知っていたとしてもだめなのだ。何が必要かを知り，設計し，アルゴリズムを考え，コーディング，デバッグする。これが問題なのである。

これから連載を始めるにしては，いきなりいーかげんであるが，これぞC調言語術の奥義なのだ。文法の勉強などというシケたことは，さっさと要領よく片づけてしまい，肝心の「このプログラム，面白いでしょ」を追及すべきなのだ。

何が面白いかということに関してはさまざまな異論があるだろうが，やっぱり，せっかくX68000なんだから，やっぱりX68000なのである。よって，本連載中にはときどきX68000でしか動かないプログラムも出没する予定である。ある人は，「Cの連載だなんて言って，本当は“試験に出るX68000”にするつもりなんじゃないですか」と言っていたが，半分当たっている。ただし，私の考えとしては，1/4 当たってるぐらいでとどめておきたいと思っている。

C概観

最近は本屋にもCの本がかなり積まれているのを見かけるようになった。極論すれば，並んでいる本はCの入門書か，さもなくば (98の) MS-DOSの入門書だと言ってもいーぐらいである。

そういえば，何年か前のことであるが，「BASICの次はPascalの時代になるのだっ。さあ来る今来る」などと真面目に論じられていた時代があった。今から思えば，どうもパソコン雑誌が，「なーるほど，Pascalねぇ。ふむふむ，そいじゃ来月号の特集は“Pascalの時代がやって来た”なんての，どう？」のノリによって，読者をおいてけぼりにしたまま大ボケを演じてしまったのではなかったかと思う。

で，現在のCブームであるが，どうやらBASICに取って代わるという感じは全然ない。その要因としては，もはやBASICはなくならないだろう，という認識がある。つまり，Cは上級者用の言語，もしくはアセンブラの代替品という考えのようである。CはBASICの上位として，別の地位を確保しつつあるのではないか。

Cの特徴

CはBASICやPascalより使い方が難しい言語である。そして実行速度はPascalより明らかに速く，また柔軟性に富んでいる。

Cがどれぐらい柔軟かという例が，リスト0である。これはどーゆープログラムかというと，暴走するプログラムなのである。ただし，X68000ではメモリにスーパーバイザ領域つーもんがあるので，このプログラムを走らせると，画面の中央にすでに皆さんには顔馴染みのことと思われる，

　アドレスエラーが発生しました

という心温まるメッセージが現れて止まってくれる。しかしほかのパソコンでは，普通はそこまで面倒は見てくれないので，やがてはメモリが0で埋め尽くされ，めでたく暴走ということになる。よーするにCという言語はこんなに簡単なプログラムで暴走してしまうくらい柔軟なのである。

そのほかにもCの特徴としては，「実行時のエラーチェックをしない」ということがある。先ほどのリスト0もそうなのだが，ここではもっと身近な例である配列の添え字の範囲のチェックを考えてみる。たとえばBASICでなら，

```
10  DIM A (10)
20  PRINT A (20)
```

を実行したならば，たちまちエラーが発生しプログラムの実行が終わるであろう。これはFortranやPascalなどのコンパイル言語でも基本的には同じである。コンパイル時には問題がなくても，いざ実行するとランタイムエラーというエラーが発生するのである（例外もあるが)。

しかしCでは，基本的にはエラーチェックはされない。であるからして，

```
char  c[10]
c[100]='A';
```

などとしてもエラーは発生せず，ただ単にメモリのどこかに'A'のアスキーコードの41Hが書き込まれるだけである。たまたまそこがプログラムの一部だとすると，もしもあな

リスト0

```
main()
{       char *p;
        for(;;)
          *p++ = 0;
}
```

たが幸運だったなら，プログラムは暴走するであろう。逆に，もしもあなたのLUCK POINTが低ければ，何度ソースプログラムを見ても発見できないバグに悩まされることになる可能性が高い。実は，Cが上級者用の言語だというのも，ここらへんに根拠がある。ただし，最近ではパソコン上でもCのデバッグ環境が整いつつあるので，エラーが発見できるようにしてコンパイルし，ちゃんと動くことが確認されたらエラーチェックなしでコンパイルするということも可能になっているよーである。

また，本当の清く正しいCでは，プログラム中でちゃんとアセンブラが使えるようになっているので（XCもそうなっている。ただし，X1のランゲージシリーズのC：αC はそうなっていないので，濁り悪しきCと言わざるを得ない。もっともそれによって，使いやすくなっているという面もあるのだが），柔軟性という点ではほかの高級言語の比ではないのである。

イントロダクションの最後に，基本方針を述べておこう。

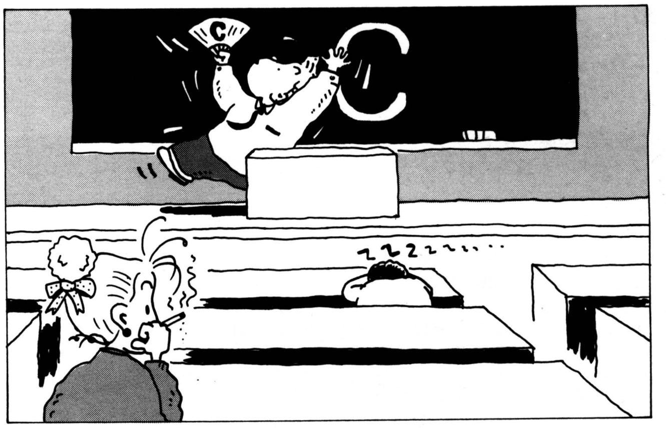
参考書はまずは前述の「K&Rの教科書」である。「K&Rの75ページ」などのようにして参照するであろう。はっきり言って，いやしくもCを使わんと欲する者は，K&Rを持ち，ひととおりは目を通しておくべきであろう。それと，当然のようにXCに付属のマニュアル群も参考書となる。また，サンプルプログラムなどは特に違うと明記していない限りPDSである。

今月のテーマ

今月はCにおける基本中の基本であるところの，printf をやるのである。で，そのために，なんとX-BASIC の外部関数を作ってしまって，BASICからprintfを使えるようにしてしまうのである（ただしちょっと制限があるが，それはあとで述べる）。Cであるからには，いかに簡単なプログラムであっても，エディタできちんとソースを作ってコンパイラにかけなければならない。そこへもってきて初心者の場合には山ほど動かないプログラムを書いてしまうわけであるから，これはとてつもなく効率が悪いのである。そんなわけで，手軽に使えるBASIC上でCの基本であるprintfを練習できれば極楽であろう。またX68000ユーザーであればXCを持っていなくても同等の練習ができるという特典もあるのでよろしく。

表1　XCの書式指定

typeの値	引　値	内　　容
%d	整数型	符号つき10進整数
%u	整数型	符号なし10進整数
%o	整数型	符号なしの8進整数
%x	整数型	符号なしの“abcdef”を使った16進数
%X	整数型	符号なしの“ABCDEF”を使った16進整数
%f	double型	double 値を［－］ddd. ddd の小数点形式に変換して出力します 小数点以下の桁数は digit により指定されます
%e	double型	double 値を［－］d. ddde［－］ddd の指数形式に変換して出力する 小数点以下の桁数は digit により指定される
%E	double型	“e”の代わりに“E”を使う以外は e フォーマットと同じ
%g	double型	double 値を，f フォーマット（小数点形式）またはeフォーマット（指数形式）のうち，結果が短くなる形式に変換して出力する 指数部が－4以下または桁数がdigitよりも大きくなる場合はeフォーマットになる
%G	double型	指数の前に“e”の代わりに“E”が使われる以外はgフォーマットと同じ
%c	文　字	1文字
%s	文字列	ヌル文字（'¥0'）で終了する文字列 または digit の数の文字列



さて，printfの“f”は「フォーマット指定付き」ということである。これはBASICでいうところのprint usingのようなもんだ。

K&Rで最初に出てくるプログラムが

```
main ( )
{
        printf ("hello, world¥n");
}
```

であることからもわかるように，このprintfはCでは基本中の基本なのである。これの詳しい説明はK&Rの158ページからとなっている。で，printf の一番簡単な形式が，先刻の，

printf（文字列）;

という形である。もっとも，これだけだったならば，たんなる文字列表示関数にすぎない。そこで，printf では，この第1引数の文字列の中に

「あとに並んでいる文字列とか数値も表示してね」

という指定を入れておくことができるようになっている。すなわち，printfの一般的な形式は，

printf（書式指定文字列，表示されるもの，……）;

となっているのだ。で，その指定は「%で始まる文字列」ということになっている。XCの場合，具体的には表1である（ほかのCでは違うこともある。Cでは，printf は必須ミネラルであるにもかかわらず，立場的にはライブラリのひとつにすぎないので，厳密な規定はないのである。K&Rに書かれている書式指定は，XCにあるものよりもずっと少ない。K&Rは最低限の要求にすぎないのだ）。

この中で一番大事なのは「%d」である。これ以外では，覚えておくべきなのは，「%c」「%s」「%f」の3つである。これ以外には「%u」が，かろうじて使うことがあるかないかであろう。

たいていはここまで知っていれば不自由することはないであろう。しかしprintfでは，もっと細かな書式指定ができるようになっている。その一般形式を表2に示しておこう。しかし，はっきり言うが，

表2に示してあることは，別に覚える必要はない

表2-1

一般的なフォーマット
%[flag][length][.digit][{l / L}]type

"%" と type 以外は省略可能。
フォーマット指定の各フィールドの内容は次のとおり。

フィールド	内容
%	書式指定の始まりを示す 省略できない
flag	記号，空白，小数点，8進，16進などの出力と表示を指定する
length	出力文字の最小桁数を指示する
digit	最小文字数，または小数点以下の桁数を指定する
l，L	引数がlong型であることを示す（XCではあってもなくても同じ）
type	変換のタイプを指定する

"%" のうしろに format_string として無意味な文字があると，そのまま出力される。

表2-3

[length]
出力する最小文字数を10進整数で指定する。
出力する文字数が length よりも少ない場合は，flag に従い右詰め，もしくは左詰めで出力する。
出力する文字数が length よりも多い場合，または length が指定されていない場合は，すべての文字を出力する。
length の先頭に "0" がついている場合，"0" で空きを埋める
length を引数で指定する場合には，アスタリスク(*)を使用する。length の引数は，フォーマット出力する値の前になければならない。

のである（覚えなくてもいいことを覚えずにすましてしまうということも大事なことなのである)。知っていて損しないのが「桁数の指定」と「空白の代わりに "0" を使う指定」であるが，これらとて知らなくても別に困ることもないだろう。

そいでもって，これはすべてに通ずることなのであるが，やはり習うより慣れろである。そこでさっさと今月のプログラムに突入するのであった。

リスト5はX-BASIC用にprintf( )を組み込んでしまうものである。しかし，170行を見るとわかるように，正確には printf(標準出力へプリントする）ではなく，cprintf（画面へプリントする）なのである。BASICには標準出力という考え方がないので(むっ) printf を呼び出すとバスエラーなどが起きるであろう。

当然このプログラムのミソは170行にある「JSR　_cprintf」なのである。BASICの外部関数の中から，Cのライブラリを呼んでも大丈夫なのかという疑問が出るかもしれないが，実際に大丈夫なのであるから，きっと大丈夫なのに違いない。この手を使えば，もっと変でオイシイこともできるかもしれないから，工夫していただきたい。ダンプリストがリスト6であるから，Cを持っていない人はシコシコと打ち込んでいただきたいのだが，はっきり言って，XCは「買い」である。最適化が甘いなどの問題があるが，BASICコンパイラも付いて39,800円，100g当たり880円というのは，かなりのハイコストパフォーマンスであろう。

てなとこで，プログラムの使い方に入るのである。

1)　ソースを，printf.sのファイル名で入力する。
2)　cc　/Zprintf.fnc printf.s

表2-2

[flag]

flagの値	内容
−	length の桁数内で左詰めで出力する 省略すると右詰めで出力する
+	出力数値が正の数のときは "+" を，負の数のときは "−" をつけて出力する 省略すると，負の数のときのみ "−" をつけ，正の数のときは符号をつけない
空白	出力数値が正の数のときは1文字分の空白を，負の数のときは "−" をつけない 省略すると，負の数のときのみ "−" をつけ，正の数のときは空白をつけない flag に "+" が指定されているときは，この指定は無効になる
#	type の値が o，x，X のとき，出力数値の先頭に0, 0x，0Xをつける 省略すると何もつけない
	type の値が f，e，E のとき，digit で指定した桁数の小数点以下の数を表示する 小数点以下に数字がないときは 0 を表示する 省略すると，小数点以下に数字があるときのみ小数点以下を表示する
	type の値が g，G のとき，digit で指定した桁数の小数点以下の数を表示し，切り捨てや余分な 0 の削除は行わない 省略すると，小数点以下に数字があるときのみ小数点以下を表示し，余分な 0 は削除する
	type の値が c，d，u，s のときは無視される

表2-4

[digit]
digit は出力する最大文字数または小数点以下の桁数を正の10進数の整数で指定する
digit を引数で指定する場合には，アスタリスク(*)を使用する。digit の引数は，フォーマット出力する値の前になければならない。
digit の指定方法，および省略した場合のデフォルト値は次のとおり。

typeの値	内容
d u o x X	出力する最小桁を指定する 引数 arg の桁数が digitより少ない場合は 0 を付加する 桁数が digitを超えても値の切り捨ては行われず，そのまま出力する 省略したとき，0 を指定したとき，またはピリオドのみが記述されたときは "1" とみなす
f e E	小数点以下の桁数を指定する 出力する最後の桁は四捨五入される 省略したときは 6 桁になる 0 を指定したとき，またはピリオドのみが記述されたときは，小数点のみを出力し，小数点以下の数字を切り捨てる
g G	出力する有効値の最大桁数を指定する 省略したときは有効値のすべてを出力する
c	無効
s	出力する最大文字数を指定する 省略したときは，文字列の末尾のヌル文字に達するまで出力する

リスト1　サンプル1

```
 10 int x0,y0
 20 x0=pos:y0=csrlin-1:print:print:print
 30 /*
 40 int i(0)
 50 char c(0)
 60 str s(0)
 70 float f(0)
 80 /*
 90 i(0)=108
100 c(0)='@'
110 s(0)="鰍鮫鰆鯵鰈"
120 f(0)=3.14159265#
130 printf("%d %c %s %f ",i,c,s,f)
140 locate x0,y0
```

リスト2　サンプル2

```
 10 str s(0)
 20 s(0)="hello, world"
 30 printf(":%10s:¥n",s)
 40 printf(":%-10s:¥n",s)
 50 printf(":%20s:¥n",s)
 60 printf(":%-20s:¥n",s)
 70 printf(":%20.10s:¥n",s)
 80 printf(":%-20.10s:¥n",s)
 90 printf(":%.10s:¥n",s)
100 locate pos,csrlin+10
```

表3　XCのエスケープ文字列

¥n	復帰改行(0aH)
¥t	水平タブ(09H)
¥v	垂直タブ(0bH)
¥b	バックスペース(08H)
¥r	復帰(0dH)
¥f	改ページ(0cH)
¥a	警告ベルを鳴らす(?)
¥'	シングルクォーテーション(27H)
¥"	ダブルクォーテーション(22H)
¥¥	円記号(5cH)
¥nnn	8進数で表されたASCII文字
¥xnnまたは¥Xnn	16進数で表されたASCII文字

リスト3　サンプル3

```
 10 int x0,y0
 20 x0=pos:y0=csrlin-1:print:print:print
 30 /*
 40 int l(0),d(0)
 50 float f(0)
 60 /*
 70 l(0)=20:d(0)=10
 80 f(0)=3.14159265#
 90 printf("%*.*f ",l,d,f)
100 locate x0,y0
```

リスト4　サンプル4

```
 10 int x0,y0
 20 x0=pos:y0=csrlin-1:print:print:print
 30 /*
 40 int i(0),j(0),k(0)
 50 /*
 60 i(0)=&H4126
 70 j(0)=&HFEF
 80 k(0)=&H88
 90 printf("%04x¥n%04x¥n%04x¥n",i,j,k)
100 locate x0,y0
```

でコンパイルする。asではないことに注意。できあがったprintf. fncはBASIC. Xと同じディレクトリにおいておくこと。

3) BASIC. CNFに，
FUNC＝PRINTF
の1行を付け加える。

これでBASICを起動すれば，printfが使えるわけだ。XC を持ってない人はダンプリストを打ち込みprintf. fnc のファイル名でセーブし同様なことをすべし。

printf の使い方であるが，形式は

printf(str[, ary1][, ary1][, ary1][, ary1][, ary1][, ary1][, ary1][, ary1][, ary1])

である。表示される引数は9個までである。このとき，

printf(s, i, , k)

のような，パラメータの飛びは許されない。

で，いよいよ具体的なサンプルがリスト1～4である。このように，いちいち配列を宣言し，それに代入してからprintf に渡さなければいけないというのは，かなりめんどくさいのであるが，困ったことにX-BASICでは，int，char，str，floatを，型にかかわらず外部関数に渡すには，この方法しかないらしいのである。

つねづね前から思っていたのだが，どうも X-BASIC というのは，ナニである。「こんなこといいな，できたらいいな」がドラえもんの4次元ポケットだとすれば，X-BASICのほうは，「低次元はらがけ」ではないかと思ったりもする。

さて，リスト1が基本型，リスト2がK&Rの159ページに載っている，文字列表示に桁数指定をした例である。注意点は，100行のlocate文である。BASICのカーソル移動は独自なので，外部関数で文字を表示すると，両者はガッチャンコして表示が乱れてしまうのである。100 行を削って走らせてみるとよくわかるであろう。

リスト3は，桁指定に「*」を使ってみた例である。

リスト4は，4桁の16進数で左側を0で埋めるようにしたものである。BASICでは，

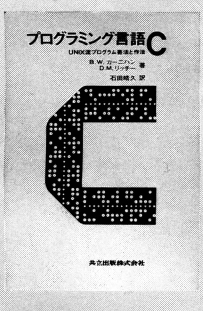
PRINT RIGHT$("000"+HEX$(I), 4)

としてやっていたものが，Cではこれですんでしまう。普通Cでは，数字の左側に0xがつくと16進数で，0だけがつくと8進数ということになっているが（つまり0x＝&H, 0だけだと&Oだ）この例における「04」は，8進数ではなく，「幅4桁でスペースの代わりに“0”を詰める」という指定なのである。

これ以上は，説明するよりも，実際にやってみてもらったほうがよいであろう。なお，引数の数や順序を間違えたり，実数を%dの指定で表示させようなどとすると，たちまち奇怪な現象に出くわすはずである。バスエラーなどが起きたとしても，BASICは壊れてはいないはずなので，そんなに心配せずにエラーしていただきたい。

最後にエスケープ文字列というやつを説明しておく。これはコントロールコードなどを，目で見えるようにするための方便である。BASICでなら，A$+CHR$（&H0D）などという表現方法があるが，Cでは文字列の足し算などはないので，こうなっているのであろう。実際BASICにも欲しいぐらいの機能である。で，XCでは，表3がサポートされている。カッコの中は文字コードである。この中で，16進数でアスキーコードを指定できるのに，10進数ではできないというのは実に奇怪である。¥dnnnなんてのは

受講生の皆さんにお知らせ



プログラミング言語C
B.W.カーニハン
D.M.リッチー著
石田晴久 訳　2,500円
共立出版　☎03(947)2511

今月から新しくスタートした「C調言語講座PRO-68K」ですが，ちょっとタイトルの祝一平氏の名前のところを見てください。「満開製作所」とありますね。そうです，察しのいい方ならお気づきでしょうが，この連載は祝氏が発行しているディスクマガジン「電脳倶楽部」に同時連載されることと相成ったのです。

さて，本文中にもあるように，この講座では「K&R」をサブテキストとして平然と引用することとなっています。なにしろすべてのCプログラマが愛読するといわれるバイブルです。間違いなくお勧めできる1冊としてご紹介しておきましょう。（編）

おいしいではないか。

今月のプログラムの書式指定文字列の中では，これらすべてはサポートしていないが，K&Rに載っている「¥n」，「¥t」，「¥'」，「¥"」，「¥¥」，「¥nnn」はサポートしてある。¥nnn があるんだから，基本的には困ることはないであろう。なお，書式指定でない文字列中では，これらのエスケープ文字はサポートされていない。すなわち，'¥n' はモロに '¥n' のままである。

最後の最後に言っておくが，どーゆーわけだか，Cでは復帰改行が$0aなのである。しかし，printfに$0a を与えても改行はしてくれるが復帰をしてくれない。それで，プログラムでは¥nを$0aではなく$0d0aに変換している（リスト5の164行～）。これはなかなか奇怪なことである。ヒマな人は調べていただきたいものである。

今月はまあまあのいーかげんさであった。ではまた来月。

リスト5 printf.s

```
  1: ****************************************
  2: *       X-BASIC EXT_FUNCTION           *
  3: *               printf()               *
  4: *                                      *
  5: *       Programmed by Ippei Iwai       *
  6: *                                      *
  7: *    Ｏｈ！Ｘ    ８８’７月号  掲載     *
  8: *    電脳倶楽部  ８８’７月号  掲載     *
  9: *                                      *
 10: *               << PDS >>              *
 11: *                                      *
 12: ****************************************
 13: *
 14: CR      EQU     $0D
 15: LF      EQU     $0A
 16: *
 17:         .TEXT
 18:         DC.L    X_INZ
 19:         DC.L    X_RUN
 20:         DC.L    X_END
 21:         DC.L    X_SYSTEM
 22:         DC.L    X_BRK
 23:         DC.L    X_CTRL_D
 24:         DC.L    X_RETADRS
 25:         DC.L    X_RETADRS
 26:         DC.L    PTR_TOKEN
 27:         DC.L    PTR_PARAM
 28:         DC.L    PTR_EXEC
 29:         DC.L    0,0,0,0,0
 30:
 31: *.......... FUNCTION NAME TABLE...........
 32: PTR_TOKEN
 33:         DC.B    'printf',0      *printf(str,ary1,ary1,...,ary1)
 34:         DC.B    0
 35:
 36:         .EVEN
 37: *....... FUNCTION PARAMETERS TABLE ......
 38: PTR_PARAM
 39:         DC.L    COM_PARAM
 40:
 41: str_val         equ     $0008
 42: any_ary1_omt    equu    $00bf   *省略可能な１次元配列へのポインタ
 43: int_ret         equ     $8001
 44:
 45: COM_PARAM
 46:         DC.W    str_val                 *FORMAT
 47:         DC.W    any_ary1_omt,any_ary1_omt
 48:         DC.W    any_ary1_omt,any_ary1_omt
 49:         DC.W    any_ary1_omt,any_ary1_omt
 50:         DC.W    any_ary1_omt,any_ary1_omt
 51:         DC.W    any_ary1_omt
 52:         DC.W    int_ret
 53:
 54: *....... FUNCTION ADDRESS TABLE ........
 55: PTR_EXEC
 56:         DC.L    b_cprintf
 57:
 58: X_INZ:
 59: X_RUN:
 60: X_END:
 61: X_SYSTEM:
 62: X_BRK:
 63: X_CTRL_D:
 64: X_RETADRS
 65:         RTS
 66: *
 67:         .TEXT
 68: b_cprintf
 69: *
 70:         MOVE.L  SP,OLD_SP       *SAVE STACK
 71:         MOVEQ.L #10-1-1,D7
 72:         LEA     96(SP),A1       *A1=TYPE OF LAST PARAM
 73: *
 74: LOOP1   TST.W   (A1)            *CHECK TYPE
 75:         BPL     LOOP2           *$FFFF == OMIT PARAMETER ?
 76:         LEA     -10(A1),A1      *A1=A1-10
 77:         DBRA    D7,LOOP1
 78: *
 79: LOOP2   MOVE.W  (A1),D0         *GET TYPE
 80:         EXT.L   D0              *SIGN EXTEND
 81:         BMI     ERROR           *IF D6==$FFFF THEN ERROR
 82:         BEQ     DO_FLOW         *IF D6==0 THEN FLOAT
 83:         SUBQ.L  #2,D0
 84:         BMI     DO_INT
 85:         BEQ     DO_CHR
 86: DO_STR
 87:         MOVE.L  6(A1),A2
 88:         PEA     10(A2)          *PUSH POINTER
 89:         LEA     -10(A1),A1      *A1-=10
 90:         DBRA    D7,LOOP2
 91:         BRA     DO_IT
 92: *
 93: DO_CHR
 94:         MOVE.L  6(A1),A2
 95:         MOVE.B  10(A2),D0       *GET VAL
 96: *
 97:         EXT.W   D0              *SIGN EXTEND
 98:         EXT.L   D0
 99: *
100:         MOVE.L  D0,-(SP)        *PUSH VAL
101:         LEA     -10(A1),A1      *A1-=10
102:         DBRA    D7,LOOP2
103:         BRA     DO_IT
104: DO_INT
105:         MOVE.L  6(A1),A2
106:         MOVE.L  10(A2),-(SP)    *PUSH VAL
107:         LEA     -10(A1),A1      *A1-=10
108:         DBRA    D7,LOOP2
109:         BRA     DO_IT
110: DO_FLOW
111:         MOVE.L  6(A1),A2
112:         MOVEM.L 10(A2),D0/D1    *GET VAL
113:         MOVEM.L D0/D1,-(SP)     *PUSH VAL
114:         LEA     -10(A1),A1      *A1-=10
115:         DBRA    D7,LOOP2
116: *
117: DO_IT
118:         MOVE.L  6(A1),A1        *A1 POINTS FORMAT
119:         LEA     FSTR,A2         *ARRANGE FORMAT
120:         MOVEQ.L #'¥',D6
121: FCONVL
122:         MOVE.B  (A1)+,D0
123:         CMP.B   D6,D0           *D0=='¥' ?
124:         BEQ     SLASH           *THEN BRANCH
125:         MOVE.B  D0,(A2)+        *GET NEXT CHAR
126:         BNE     FCONVL          *NOT NULL THEN LOOP
127:         BRA     GO_PRINTF       *DO CPRINTF
128: *
129: SLASH   MOVE.B  (A1)+,D0        *GET NEXT CHAR
130:         CMP.B   #'n',D0         *'¥n' ?
131:         BEQ     DO_N
132:         MOVEQ.L #$09,D1         *'¥t' ?
133:         CMP.B   #'t',D0
134:         BEQ     SEND
135:         CMP.B   #'7',D0         *'¥ddd' ?
136:         BGT     SEND0
137:         CMP.B   #'0',D0
138:         BLT     SEND0
139: *OCTAL NUMBER
140:         MOVEQ.L #0,D1
141:         MOVEQ.L #3-1,D2         *COUNTER (MAX 3 KETA)
142: *
143: OCTLOOP
144:         SUB.B   #'0',D0
145:         ASL.B   #3,D1
146:         ADD.B   D0,D1
147:         MOVE.B  (A1)+,D0
148:         CMP.B   #'7',D0
149:         BGT     OCTEND          *NOT OCTAL THEN END
150:         CMP.B   #'0',D0
151:         BLT     OCTEND          *NOT OCTAL THEN END
152:         DBRA    D2,OCTLOOP      *CHECK KETA
153: OCTEND
154:         MOVE.B  D1,(A2)+
155:         SUBQ.L  #1,A1           *A1=A1-1
156:         BRA     FCONVL
157:
158: SEND0   MOVE.B  D0,(A2)+
159:         BRA     FCONVL
160:
161: SEND    MOVE.B  D1,(A2)+
162:         BRA     FCONVL
163:
164: DO_N    MOVE.B  #CR,(A2)+
165:         MOVE.B  #LF,(A2)+
166:         BRA     FCONVL
167:
168: GO_PRINTF
169:         PEA     FSTR
170:         JSR     _cprintf
171:         MOVE.L  OLD_SP,SP       *BACK SP
172:         MOVE.L  D0,INT_DATA     *RETURN VALUE
173:         LEA     RET_DATA,A0
174:         CLR.L   D0              *NO ERROR
175:         RTS
176: *
177: ERROR
178:         MOVE.L  OLD_SP,SP       *BACK SP
179:         MOVEQ.L #1,D0
180:         LEA     MES0,A1
181:         RTS
182: *
183: OLD_SP  DS.L    1
184: *
185: RET_DATA
186:         DS.W    1
187:         DS.L    1
188: INT_DATA
189:         DS.L    1
190:
191: MES0:   DC.B    'パラメータの並びに抜けている部分があります',0
192: *
193: FSTR    DS.B    256
194: *
195:         .END
```

リスト6 printf.fnc

```
0000  48 55 00 00 00 00 00 00  : 9D
0008  00 00 00 00 00 00 0D 9A  : A7
0010  00 00 00 00 00 00 00 00  : 00
0018  00 00 00 46 00 00 00 46  : 8C
0020  00 00 00 00 00 00 00 00  : 00
0028  00 00 00 00 00 00 00 00  : 00
0030  00 00 00 00 00 00 00 00  : 00
0038  00 00 00 00 00 00 00 00  : 00
0040  00 00 00 66 00 00 00 66  : CC
0048  00 00 00 66 00 00 00 66  : CC
0050  00 00 00 66 00 00 00 66  : CC
0058  00 00 00 66 00 00 00 66  : CC
0060  00 00 00 40 00 00 00 48  : 88
0068  00 00 00 62 00 00 00 00  : 62
0070  00 00 00 00 00 00 00 00  : 00
0078  00 00 00 00 00 00 00 00  : 00
-----------------------------------
SUM:  48 55 00 80 00 00 0D C0  367F

0080  70 72 69 6E 74 66 00 00  : 93
0088  00 00 00 4C 00 08 00 BF  : 13
0090  00 BF 00 BF 00 BF 00 BF  : FC
0098  00 BF 00 BF 00 BF 00 BF  : FC
00A0  80 01 00 00 00 68 4E 75  : AC
00A8  23 CF 00 00 01 7E 7E 08  : F7
00B0  43 EF 00 60 4A 51 6A 08  : 9F
00B8  43 E9 FF F6 51 CF FF F6  : 36
00C0  30 11 48 C0 6B 00 00 E8  : 9C
00C8  67 42 55 80 6B 2C 67 12  : 8E
00D0  24 69 00 06 48 6A 00 0A  : 4F
00D8  43 E9 FF F6 51 CF FF E2  : 22
00E0  60 40 24 69 00 06 10 2A  : 6D
00E8  00 0A 48 80 48 C0 2F 00  : 09
00F0  43 E9 FF F6 51 CF FF CA  : 0A
00F8  60 28 24 69 00 06 2F 2A  : 74
-----------------------------------
SUM:  9A 98 93 12 18 F2 08 BC  3496

0100  00 0A 43 E9 FF F6 51 CF  : 4B
0108  FF B8 60 16 24 69 00 06  : C0
0110  4C EA 00 03 00 0A 48 E7  : 72
0118  C0 00 43 E9 FF F6 51 CF  : 01
0120  FF A0 22 69 00 06 45 F9  : 6E
0128  00 00 01 B7 7C 5C 10 19  : B9
0130  B0 06 67 06 14 C0 66 F6  : 53
0138  60 52 10 19 B0 3C 00 6E  : 35
0140  67 40 72 09 B0 3C 00 74  : 82
0148  67 34 B0 3C 00 37 6E 2A  : 56
0150  B0 3C 00 30 6D 24 72 00  : 1F
0158  74 02 90 3C 00 30 E7 01  : 5A
0160  D2 00 10 19 B0 3C 00 37  : 1E
0168  6E 0A B0 3C 00 30 6D 04  : 05
0170  51 CA FF E8 14 C1 53 89  : B3
0178  60 B4 14 C0 60 B0 14 C1  : CD
-----------------------------------
SUM:  FD DE 05 D8 A3 61 40 25  B225

0180  60 AC 14 FC 00 0D 14 FC  : 39
0188  00 0A 60 A2 48 79 00 00  : CD
0190  01 B7 4E B9 00 00 02 B8  : 79
0198  2E 79 00 00 01 7E 23 C0  : 09
01A0  00 00 01 88 41 F9 00 00  : C3
01A8  01 82 42 80 4E 75 2E 79  : AF
01B0  00 00 01 7E 70 01 43 F9  : 2C
01B8  00 00 01 8C 4E 75 00 00  : 50
01C0  00 00 00 00 00 00 00 00  : 00
01C8  00 00 00 00 83 70 83 89  : FF
01D0  83 81 81 5B 83 5E 82 CC  : 0F
01D8  95 C0 82 D1 82 C9 94 B2  : 39
01E0  82 AF 82 C4 82 A2 82 E9  : 06
01E8  95 94 95 AA 82 AA 82 A0  : B6
01F0  82 E8 82 DC 82 B7 00 00  : 01
01F8  00 00 00 00 00 00 00 00  : 00
-----------------------------------
SUM:  41 D4 A3 DF A4 82 47 76  2611

0200  00 00 00 00 00 00 00 00  : 00
0208  00 00 00 00 00 00 00 00  : 00
0210  00 00 00 00 00 00 00 00  : 00
0218  00 00 00 00 00 00 00 00  : 00
0220  00 00 00 00 00 00 00 00  : 00
0228  00 00 00 00 00 00 00 00  : 00
0230  00 00 00 00 00 00 00 00  : 00
0238  00 00 00 00 00 00 00 00  : 00
0240  00 00 00 00 00 00 00 00  : 00
0248  00 00 00 00 00 00 00 00  : 00
0250  00 00 00 00 00 00 00 00  : 00
0258  00 00 00 00 00 00 00 00  : 00
0260  00 00 00 00 00 00 00 00  : 00
0268  00 00 00 00 00 00 00 00  : 00
0270  00 00 00 00 00 00 00 00  : 00
0278  00 00 00 00 00 00 00 00  : 00
-----------------------------------
SUM:  00 00 00 00 00 00 00 00  0000

0280  00 00 00 00 00 00 00 00  : 00
0288  00 00 00 00 00 00 00 00  : 00
0290  00 00 00 00 00 00 00 00  : 00
0298  00 00 00 00 00 00 00 00  : 00
02A0  00 00 00 00 00 00 00 00  : 00
02A8  00 00 00 00 00 00 00 00  : 00
02B0  00 00 00 00 00 00 00 00  : 00
02B8  00 00 00 00 00 00 00 00  : 00
02C0  00 00 00 00 00 00 00 00  : 00
02C8  00 00 00 00 00 00 00 00  : 00
02D0  00 00 00 00 00 00 00 00  : 00
02D8  00 00 00 00 00 00 00 00  : 00
02E0  00 00 00 00 00 00 00 00  : 00
02E8  00 00 00 00 00 00 00 00  : 00
02F0  00 00 00 00 00 00 00 00  : 00
02F8  48 6F 00 08 2F 2F 00 08  : 25
-----------------------------------
SUM:  48 6F 00 08 2F 2F 00 08  718B

0300  42 A7 48 79 00 00 02 D4  : 80
0308  4E B9 00 00 02 E2 4F EF  : 29
0310  00 10 4E 75 20 2F 00 04  : 26
0318  3F 00 FF 02 42 80 30 1F  : 51
0320  4E 75 48 E7 1F 1E 4F EF  : 6D
0328  FD 94 2C 6F 02 94 2A 6F  : 5B
0330  02 A0 28 6F 02 9C 2C 2F  : 32
0338  02 98 42 87 42 84 42 85  : F0
0340  26 4F 61 12 08 04 00 19  : 0D
0348  67 F2 4F EF 02 6C 20 07  : 2C
0350  4C DF 78 F8 4E 75 10 1C  : 8A
0358  66 06 08 C4 00 19 60 10  : C1
0360  24 4C 0C 00 00 25 66 04  : 0B
0368  61 08 60 04 61 00 06 06  : 3A
0370  4E 75 10 1C 66 06 08 C4  : 27
0378  00 19 4E 75 0C 00 00 2D  : 15
-----------------------------------
SUM:  30 B9 6D 8E F4 8C 6C 3F  F501

0380  66 06 08 C4 00 1F 60 EA  : A1
0388  0C 00 00 2B 66 06 08 C4  : 6F
0390  00 1E 60 DE 0C 00 00 20  : 88
0398  66 06 08 C4 00 16 60 D2  : 80
03A0  0C 00 00 23 66 06 08 C4  : 67
03A8  00 17 60 C6 76 FF 0C 00  : BE
03B0  00 30 66 06 08 C4 00 1A  : 82
03B8  60 12 0C 00 00 2A 66 0C  : 1A
03C0  22 1D 36 01 10 1C 67 00  : 09
03C8  05 A6 60 18 B0 3C 00 30  : 3F
03D0  65 12 B0 3C 00 39 62 0C  : 0A
03D8  61 00 05 B4 6B 00 05 90  : 1A
03E0  36 00 48 40 0C 00 00 2E  : F8
03E8  66 3C 08 C4 00 1D 10 1C  : B7
03F0  67 00 05 7C 0C 00 00 2A  : 1E
03F8  66 10 22 1D 48 43 36 01  : 77
-----------------------------------
SUM:  9A A4 04 26 E1 1F 56 CB  A5D1

0400  48 43 10 1C 67 00 05 68  : 8B
0408  60 1C B0 3C 00 30 65 16  : 13
0410  B0 3C 00 39 62 10 61 00  : F8
0418  05 76 6B 00 05 52 48 43  : C8
0420  36 00 48 43 48 40 0C 00  : 55
0428  00 6C 67 06 0C 00 00 4C  : 31
0430  66 0A 08 C4 00 1C 10 1C  : 84
0438  67 00 05 34 0C 00 00 41  : ED
0440  65 00 05 2C 0C 00 00 5A  : FC
0448  62 0A 08 C4 00 1B D0 3C  : 5F
0450  00 20 60 10 0C 00 00 61  : FD
0458  65 00 05 14 0C 00 00 7A  : 04
0460  62 00 05 0C 0C 00 00 62  : E1
0468  65 00 05 04 0C 00 00 63  : DD
0470  65 00 03 80 0C 00 00 64  : 58
0478  65 00 02 A6 0C 00 00 65  : 7E
-----------------------------------
SUM:  1D B1 68 1C 82 09 FF 69  74C3

0480  65 00 03 02 0C 00 00 68  : DE
0488  65 24 0C 00 00 6F 67 00  : 6B
0490  03 6A 0C 00 00 73 67 00  : 53
0498  02 AC 0C 00 00 75 67 00  : 96
04A0  02 E4 0C 00 00 78 67 00  : D1
04A8  03 5A 60 00 04 C2 42 82  : 47
04B0  14 00 20 1D 22 1D 0C 02  : 9E
04B8  00 67 67 00 00 96 0C 02  : 72
04C0  00 66 67 4C B6 7C FF FF  : 49
04C8  66 04 36 3C 00 01 48 43  : 68
04D0  B6 7C FF FF 66 04 36 3C  : 0C
04D8  00 06 34 03 48 43 61 00  : 29
04E0  01 12 61 00 01 54 41 EB  : F5
04E8  00 01 0C 10 00 2D 67 00  : B1
04F0  02 20 08 04 00 1E 66 10  : C2
04F8  08 04 00 16 67 00 02 12  : 9D
-----------------------------------
SUM:  0F 02 5F D3 FE A7 E4 79  8157

0500  11 3C 00 20 60 00 02 0A  : D9
0508  11 3C 00 2B 60 00 02 02  : DC
0510  B6 7C FF FF 66 04 36 3C  : 0C
0518  00 01 48 43 B6 7C FF FF  : BC
0520  66 04 36 3C 00 06 34 03  : 19
0528  48 43 51 8F 20 4F 43 EF  : 0C
0530  00 04 2F 09 2F 08 2F 02  : A4
0538  48 E7 C0 00 4E B9 00 00  : F6
0540  0B 0E 4F EF 00 14 4C DF  : 96
0548  00 06 20 40 61 00 01 30  : F8
0550  60 94 B6 7C FF FF 66 04  : 8E
0558  36 3C 00 01 48 43 B6 7C  : 30
0560  FF FF 66 04 36 3C 00 0E  : E8
0568  34 03 48 43 48 E7 E0 00  : D1
0570  51 8F 20 4F 43 EF 00 04  : 85
0578  2F 09 2F 08 2F 02 48 E7  : CF
-----------------------------------
SUM:  22 A5 DF AB 11 00 70 C3  FB0D

0580  C0 00 4E B9 00 00 0B 0E  : E0
0588  4F EF 00 14 4C DF 00 06  : 83
0590  20 40 53 81 B2 7C FF FC  : 5D
0598  6D 08 20 03 48 40 B2 40  : 12
05A0  6F 10 4C DF 00 07 61 4A  : 5C
05A8  61 18 61 00 00 8C 60 00  : C6
05B0  FF 36 52 81 4F EF 00 0C  : 52
05B8  61 00 00 C4 61 04 60 00  : EA
05C0  FF 26 08 04 00 17 66 28  : D6
05C8  42 11 41 EB 00 01 10 18  : A8
05D0  67 1E B0 3C 00 2E 66 F6  : FB
05D8  10 21 B0 3C 00 30 67 F8  : AC
05E0  B0 3C 00 2E 66 06 08 04  : 92
05E8  00 17 67 02 52 89 42 11  : AE
05F0  4E 75 52 42 51 8F 20 4F  : A6
05F8  43 EF 00 04 2F 09 2F 08  : A5
-----------------------------------
SUM:  C5 C2 22 52 2E BE B9 40  1861

0600  2F 02 48 E7 C0 00 4E B9  : 27
0608  00 00 09 B8 4C DF 00 06  : F2
0610  4F EF 00 0C 82 82 4C DF  : 79
0618  00 06 66 02 72 01 20 40  : 41
0620  43 EB 00 01 4A 82 67 04  : 66
0628  12 FC 00 2D 61 00 00 D6  : 72
0630  12 C0 61 00 00 A6 4E 75  : 9C
0638  10 3C 00 65 08 04 00 1B  : D8
0640  67 04 10 3C 00 45 12 C0  : CE
0648  10 3C 00 2B 53 81 4A 81  : 16
0650  6A 06 10 3C 00 2D 44 81  : AE
0658  12 C0 82 FC 00 64 D2 3C  : C2
0660  00 30 12 C1 42 41 48 41  : 0F
0668  82 FC 00 0A D2 3C 00 30  : C6
0670  12 C1 48 41 D2 3C 00 30  : 9A
0678  12 C1 42 11 4E 75 43 EB  : 17
-----------------------------------
SUM:  8E 8E 56 FC 3A 13 6C D2  149F

0680  00 01 4A 82 67 04 12 FC  : 46
0688  00 2D 4A 81 6B 12 67 10  : EC
0690  53 81 61 70 12 C0 51 C9  : 91
0698  FF FA 61 3E 42 11 4E 75  : AE
06A0  44 81 20 03 48 40 12 FC  : 7E
06A8  00 30 08 04 00 17 66 04  : BD
06B0  4A 40 67 22 12 FC 00 2E  : 4F
06B8  4A 40 67 1A 53 41 6B 0C  : 16
06C0  12 FC 00 30 53 40 67 0E  : 46
06C8  51 C9 FF F6 22 00 61 34  : C6
06D0  12 C0 53 41 66 F8 42 11  : 17
06D8  4E 75 20 03 48 40 08 04  : 7A
06E0  00 17 66 04 4A 40 6F 1A  : 94
06E8  12 FC 00 2E 53 40 6B 12  : 4C
06F0  48 E7 40 00 22 00 61 0C  : FE
06F8  12 C0 51 C9 FF FA 4C DF  : 10
-----------------------------------
SUM:  59 8E B5 59 B4 6D 94 F2  A18C

0700  00 02 4E 75 10 18 66 06  : 59
0708  10 3C 00 30 53 88 4E 75  : 1A
0710  22 48 4A 19 66 FC 2A 09  : 62
0718  9A 88 53 85 60 00 01 9E  : F9
0720  B6 7C FF FF 66 04 36 3C  : 0C
0728  00 01 48 43 B6 7C FF FF  : BC
0730  66 04 36 3C 00 01 48 43  : 68
```

```
0738  41 ED 00 03 4A 9D 7A 01  : 93
0740  60 00 01 7A 20 1D 6B 0A  : 8D
0748  66 0E 20 3C 00 00 09 A8  : 81
0750  60 06 20 3C 00 00 09 AF  : 7A
0758  20 40 22 48 4A 19 66 FC  : 8F
0760  2A 09 9A 88 53 85 B6 7C  : 5F
0768  FF FF 66 02 42 43 48 43  : 76
0770  B6 7C FF FF 66 02 36 05  : D3
0778  BA 43 65 02 3A 03 48 43  : 2C
-----------------------------------
SUM:  08 97 2F 89 2E BD 35 05  BEB9

0780  60 00 01 3A 2F 0B 4F EF  : 13
0788  FF DE 20 4F 22 1D 42 85  : 52
0790  0C 00 00 75 67 24 4A 81  : D7
0798  6B 18 08 04 00 1E 66 0C  : 1F
07A0  08 04 00 16 67 14 16 FC  : AF
07A8  00 20 60 0C 16 FC 00 2B  : C9
07B0  60 06 16 FC 00 2D 44 81  : 6A
07B8  52 85 4A 81 67 2E 43 F9  : 73
07C0  00 00 09 78 20 19 B2 80  : EC
07C8  65 FA 74 2F 52 02 92 80  : 68
07D0  64 FA D2 80 10 C2 20 19  : BB
07D8  66 F0 42 10 20 4F 72 FF  : 88
07E0  52 41 4A 18 66 FA 20 4F  : C4
07E8  60 00 00 96 10 FC 00 30  : 32
07F0  60 E8 22 3C 00 01 00 01  : A8
07F8  60 0E 22 3C 00 03 00 07  : D6
-----------------------------------
SUM:  31 C0 08 FE B4 FB D4 41  A284

0800  60 06 22 3C 00 04 00 0F  : D7
0808  48 E7 00 10 42 85 08 84  : 92
0810  00 17 67 2A 16 FC 00 30  : EA
0818  52 45 0C 41 00 01 66 06  : 51
0820  16 FC 00 62 60 0A 0C 41  : 2B
0828  00 0F 66 12 16 FC 00 78  : 11
0830  52 45 08 04 00 1B 67 06  : 2B
0838  04 2B 00 20 FF FF 20 4F  : BC
0840  4F EF FF DE 20 1D 42 20  : BA
0848  24 00 48 E7 04 00 42 85  : 1E
0850  52 85 C4 01 0C 02 00 09  : B3
0858  6E 06 D4 3C 00 30 60 0E  : 22
0860  D4 3C 00 37 08 04 00 1B  : 6E
0868  66 04 D4 3C 00 20 11 02  : AD
0870  48 41 E2 A8 48 41 24 00  : C0
0878  66 D6 22 05 4C DF 00 20  : AE
-----------------------------------
SUM:  81 95 BA 71 99 39 1A D0  B561

0880  B6 7C FF FF 66 04 36 3C  : 0C
0888  00 01 48 43 B6 7C FF FF  : BC
0890  66 04 36 3C 00 01 30 03  : 10
0898  48 43 90 41 6F 0C 53 40  : 6A
08A0  16 FC 00 30 52 85 51 C8  : 32
08A8  FF F8 52 85 16 D8 66 FA  : 1C
08B0  53 85 4F EF 00 22 4C DF  : 63
08B8  08 00 20 4B BA 43 65 02  : D7
08C0  36 05 08 04 00 1F 66 64  : 30
08C8  08 04 00 1A 67 4A 43 F9  : 13
08D0  00 00 09 A5 10 10 B0 3C  : BA
08D8  00 30 67 1A B0 3C 00 20  : BD
08E0  67 0C 0C 00 00 2B 67 06  : 17
08E8  0C 00 00 2D 66 30 61 5C  : 8C
08F0  4A 18 53 45 60 28 10 28  : BA
08F8  00 01 80 3C 00 20 B0 3C  : C9
-----------------------------------
SUM:  CF 9B 25 39 9A A7 01 A0  3A86

0900  00 62 67 06 B0 3C 00 78  : 33
0908  66 14 61 40 4A 18 53 45  : 15
0910  61 3A 4A 18 53 45 60 06  : FB
0918  43 F9 00 00 09 A4 BA 43  : E6
0920  64 0A C3 48 61 26 BA 43  : FD
0928  65 FA C3 48 4A 45 67 0C  : 6C
0930  61 1A 4A 18 4A 43 6F 12  : EB
0938  53 45 66 F0 4A 43 6F 0A  : F4
0940  41 F9 00 00 09 A4 61 04  : 4C
0948  66 FC 4E 75 48 E7 7D FE  : CF
0950  10 10 02 80 00 00 00 FF  : A1
0958  48 E7 82 00 4E 96 4C DF  : C0
0960  00 41 4C DF 7F BE 70 01  : 1A
0968  DE 80 96 40 4E 75 42 84  : BD
0970  28 4A 10 1C 41 EC FF FF  : C9
0978  10 1C 67 06 0C 00 00 25  : CA
-----------------------------------
SUM:  9C 1F 73 2C 4E 6E 47 FA  1550

0980  66 F6 4A 24 2A 0C 9A 88  : 22
0988  36 05 60 00 FF 30 42 81  : 8D
0990  0C 00 00 30 65 1A 0C 00  : C7
0998  00 39 62 14 02 40 00 0F  : 00
09A0  D2 41 D0 41 E5 49 D2 40  : 64
09A8  10 1C 66 E4 72 FF 4E 75  : AA
09B0  48 40 30 01 42 81 4E 75  : 3F
09B8  3B 9A CA 00 05 F5 E1 00  : 7A
09C0  00 98 96 80 00 0F 42 40  : 3F
09C8  00 01 86 A0 00 00 27 10  : 5E
09D0  00 00 03 E8 00 00 00 64  : 4F
09D8  00 00 00 0A 00 00 00 01  : 0B
09E0  00 00 00 00 20 30 0D 00  : 5D
09E8  28 4E 55 4C 4C 29 00 28  : B4
09F0  45 52 52 4F 52 29 00 00  : B3
09F8  4C EF 06 07 00 04 20 7C  : E8
-----------------------------------
SUM:  C6 93 08 42 EC E9 CD 9B  FB4F

0A00  00 00 09 D8 42 10 B4 BC  : A3
0A08  00 00 01 35 64 06 FE 24  : C2
0A10  22 80 24 81 20 08 4E 75  : 32
0A18  00 00 00 00 00 00 00 00  : 00
0A20  00 00 00 00 00 00 00 00  : 00
0A28  00 00 00 00 00 00 00 00  : 00
0A30  00 00 00 00 00 00 00 00  : 00
0A38  00 00 00 00 00 00 00 00  : 00
0A40  00 00 00 00 00 00 00 00  : 00
0A48  00 00 00 00 00 00 00 00  : 00
0A50  00 00 00 00 00 00 00 00  : 00
0A58  00 00 00 00 00 00 00 00  : 00
0A60  00 00 00 00 00 00 00 00  : 00
0A68  00 00 00 00 00 00 00 00  : 00
0A70  00 00 00 00 00 00 00 00  : 00
0A78  00 00 00 00 00 00 00 00  : 00
-----------------------------------
SUM:  22 80 2E 8E C6 1E 00 55  BD6E

0A80  00 00 00 00 00 00 00 00  : 00
0A88  00 00 00 00 00 00 00 00  : 00
0A90  00 00 00 00 00 00 00 00  : 00
0A98  00 00 00 00 00 00 00 00  : 00
0AA0  00 00 00 00 00 00 00 00  : 00
0AA8  00 00 00 00 00 00 00 00  : 00
0AB0  00 00 00 00 00 00 00 00  : 00
0AB8  00 00 00 00 00 00 00 00  : 00
0AC0  00 00 00 00 00 00 00 00  : 00
0AC8  00 00 00 00 00 00 00 00  : 00
0AD0  00 00 00 00 00 00 00 00  : 00
0AD8  00 00 00 00 00 00 00 00  : 00
0AE0  00 00 00 00 00 00 00 00  : 00
0AE8  00 00 00 00 00 00 00 00  : 00
0AF0  00 00 00 00 00 00 00 00  : 00
0AF8  00 00 00 00 00 00 00 00  : 00
-----------------------------------
SUM:  00 00 00 00 00 00 00 00  0000

0B00  00 00 00 00 00 00 00 00  : 00
0B08  00 00 00 00 00 00 00 00  : 00
0B10  00 00 00 00 00 00 00 00  : 00
0B18  00 00 00 00 00 00 00 00  : 00
0B20  00 00 00 00 00 00 00 00  : 00
0B28  00 00 00 00 00 00 00 00  : 00
0B30  00 00 00 00 00 00 00 00  : 00
0B38  00 00 00 00 00 00 00 00  : 00
0B40  00 00 00 00 00 00 00 00  : 00
0B48  00 00 00 00 00 00 4C EF  : 3B
0B50  06 07 00 04 41 F9 00 00  : 4B
0B58  0B 2E 42 10 B4 BC 00 00  : FB
0B60  01 35 64 06 FE 25 22 80  : 65
0B68  24 81 20 08 4E 75 00 00  : 90
0B70  00 00 00 00 00 00 00 00  : 00
0B78  00 00 00 00 00 00 00 00  : 00
-----------------------------------
SUM:  36 EB C6 22 41 4F 6E 6F  623A

0B80  00 00 00 00 00 00 00 00  : 00
0B88  00 00 00 00 00 00 00 00  : 00
0B90  00 00 00 00 00 00 00 00  : 00
0B98  00 00 00 00 00 00 00 00  : 00
0BA0  00 00 00 00 00 00 00 00  : 00
0BA8  00 00 00 00 00 00 00 00  : 00
0BB0  00 00 00 00 00 00 00 00  : 00
0BB8  00 00 00 00 00 00 00 00  : 00
0BC0  00 00 00 00 00 00 00 00  : 00
0BC8  00 00 00 00 00 00 00 00  : 00
0BD0  00 00 00 00 00 00 00 00  : 00
0BD8  00 00 00 00 00 00 00 00  : 00
0BE0  00 00 00 00 00 00 00 00  : 00
0BE8  00 00 00 00 00 00 00 00  : 00
0BF0  00 00 00 00 00 00 00 00  : 00
0BF8  00 00 00 00 00 00 00 00  : 00
-----------------------------------
SUM:  00 00 00 00 00 00 00 00  0000

0C00  00 00 00 00 00 00 00 00  : 00
0C08  00 00 00 00 00 00 00 00  : 00
0C10  00 00 00 00 00 00 00 00  : 00
0C18  00 00 00 00 00 00 00 00  : 00
0C20  00 00 00 00 00 00 00 00  : 00
0C28  00 00 00 00 00 00 00 00  : 00
0C30  00 00 00 00 00 00 00 00  : 00
0C38  00 00 00 00 00 00 00 00  : 00
0C40  00 00 00 00 00 00 00 00  : 00
0C48  00 00 00 00 00 00 00 00  : 00
0C50  00 00 00 00 00 00 00 00  : 00
0C58  00 00 00 00 00 00 00 00  : 00
0C60  00 00 00 00 00 00 00 00  : 00
0C68  00 00 00 00 00 00 00 00  : 00
0C70  00 00 00 00 00 00 00 00  : 00
0C78  00 00 00 00 00 00 00 00  : 00
-----------------------------------
SUM:  00 00 00 00 00 00 00 00  0000

0C80  00 00 00 00 00 00 00 00  : 00
0C88  00 00 00 00 00 00 00 00  : 00
0C90  00 00 00 00 00 00 00 00  : 00
0C98  00 00 00 00 00 00 00 00  : 00
0CA0  00 00 00 00 00 00 00 00  : 00
0CA8  00 00 00 00 00 00 00 00  : 00
0CB0  00 00 00 00 00 00 00 00  : 00
0CB8  00 00 00 00 00 00 00 00  : 00
0CC0  00 00 00 00 00 00 00 00  : 00
0CC8  00 00 00 00 00 00 00 00  : 00
0CD0  00 00 00 00 00 00 00 00  : 00
0CD8  00 00 00 00 00 00 00 00  : 00
0CE0  00 00 00 00 00 00 00 00  : 00
0CE8  00 00 00 00 00 00 00 00  : 00
0CF0  00 00 00 00 00 00 00 00  : 00
0CF8  00 00 00 00 00 00 00 00  : 00
-----------------------------------
SUM:  00 00 00 00 00 00 00 00  0000

0D00  00 00 00 00 00 00 00 00  : 00
0D08  00 00 00 00 00 00 00 00  : 00
0D10  00 00 00 00 00 00 00 00  : 00
0D18  00 00 00 00 00 00 00 00  : 00
0D20  00 00 00 00 00 00 00 00  : 00
0D28  00 00 00 00 00 00 00 00  : 00
0D30  00 00 00 00 00 00 00 00  : 00
0D38  00 00 00 00 00 00 00 00  : 00
0D40  00 00 00 00 00 00 00 00  : 00
0D48  00 00 00 00 00 00 00 00  : 00
0D50  00 00 00 00 00 00 00 00  : 00
0D58  00 00 00 00 00 00 00 00  : 00
0D60  00 00 00 00 00 00 00 00  : 00
0D68  00 00 00 00 00 00 00 00  : 00
0D70  00 00 00 00 00 00 00 00  : 00
0D78  00 00 00 00 00 00 00 00  : 00
-----------------------------------
SUM:  00 00 00 00 00 00 00 00  0000

0D80  00 00 00 00 00 00 00 00  : 00
0D88  00 00 00 00 00 00 00 00  : 00
0D90  00 00 00 00 00 00 00 00  : 00
0D98  00 00 00 00 00 00 00 00  : 00
0DA0  00 00 00 00 00 00 00 00  : 00
0DA8  00 00 00 00 00 00 00 00  : 00
0DB0  00 00 00 00 00 00 00 00  : 00
0DB8  00 00 00 00 00 00 00 00  : 00
0DC0  00 00 00 00 00 00 00 00  : 00
0DC8  00 00 00 00 00 00 00 00  : 00
0DD0  00 00 00 00 00 00 00 00  : 00
0DD8  00 00 00 00 00 04 00 04  : 08
0DE0  00 04 00 04 00 04 00 04  : 10
0DE8  00 04 00 04 00 04 00 04  : 10
0DF0  00 20 00 1A 00 08 00 7E  : C0
0DF8  00 66 00 06 00 06 00 06  : 78
-----------------------------------
SUM:  00 8E 00 28 00 1A 00 90  3F92

0E00  00 06 00 0A 00 08 01 4C  : 65
0E08  00 06 02 34 00 46 00 84  : 06
0E10  01 44 00 08 00 6C 01 10  : CA
0E18  00 4A 00 28 00 BE 01 56  : 87
0E20  02 01 00 00 02 B8 5F 63  : 7F
0E28  70 72 69 6E 74 66 00 00  : 93
0E30  02 01 00 00 02 D4 5F 70  : A8
0E38  75 74 63 68 00 00 02 01  : B7
0E40  00 00 02 E2 5F 5F 66 6D  : 75
0E48  74 6F 75 74 00 00 02 01  : CF
0E50  00 00 09 B8 5F 65 63 76  : 5E
0E58  74 00 02 01 00 00 0B 0E  : 90
0E60  5F 66 63 76 74 00 00 00  : 12
0E68  00 00 00 00 00 00 00 00  : 00
0E70  00 00 00 00 00 00 00 00  : 00
0E78  00 00 00 00 00 00 00 00  : 00
-----------------------------------
SUM:  31 57 B3 C9 AA 2E 99 FC  EA3D
```

X68000 BASIC入門 第12回

無限作曲・MML伝説(ミュージック応用編)

Nakamori Akira
中森 章

先月お届けしたミュージック基礎攻略編に続いて，今回はその応用編としてサンプルプログラムをバシバシ用意し，そこからさらにそれらの集大成として自動作曲プログラムへと侵攻していきます。ここまで制覇すれば，もうあなたはきっと立派な作曲家(?)です。

先月に引き続いて今月もFM音源を扱ってみたいと思います。先月はMMLの解説をメインとしたいわば基礎編でした。ですから，なにかマニュアルを読んでいるみたいで，思わず眠気に襲われた人もいるのではないでしょうか。今月は眠気を吹き飛ばすためにプログラムをどんどん作って，アクティブにやりたいと思います。しかし，もしかしてどこかに細かいミスがあるやもしれません。しかしそれは，ここまで育ってきたこの私の音楽センスの末路だと思って，「実際に音が鳴る」という点を考慮しながら，プログラムの構造を追うことを優先させてくださいね(うう，音楽は苦手じゃ)。

音色を変える

パソコン雑誌に掲載されているいろいろな音楽プログラムを見ると，音色としてプリセット音をそのまま使わず，新たに設定し直している例がかなりあります。やはり，音楽に凝っている人は音色にも凝るものなのでしょう。そういえば，清水和人さんも「音色を語らずして音楽に未来はない」なんていってましたっけ(3月号参照)。というわけで，まずは音色を変えて音楽演奏をしてみましょう。

1) 音色を変更する関数

X-BASICで音色を変更する(というか追加する）関数はm_vsetです。この関数は引数として，OPMにセットすべき音色の情報を持ちます。先月で解説したとおり，OPMは4つのオペレータ（サイン波発生器）を組み合わせていろいろな波形を作り，それにエンベロープの情報やLFO(低周波発振器）の情報が合わさって，ひとつの音色を作ります。m_vset関数ではこれらの情報をひとつの5×11の2次元配列に入れて，その配列の名前を引数とします。配列の内容は図1のようになっています。

また，図1をOPMに対する機能という観点から見たものが図2，図1をOPMの資源という観点から見たものが図3です。それぞれの要素の詳しい説明は先月してありますのでここでは省略します。それより，OPM の内部レジスタの表が手元にあれば(たとえば単行本『試験に出るX1』など)，それを見てみましょう。m_vset関数の引数に与える配列の内容が，OPMのほとんどすべての内部レジスタに対応していることがわかります。あとに残っているレジスタはノイズ関係のレジスタ（あまり使われない）と，音階（ドレミ……）を指定するレジスタ（これはMMLの音符の記述によって設定される）ぐらいですから，音色の設定によってFM音源のすべてが決まってしまうといっても過言ではありません。音にこだわる人が自分で音色を設定しなければ気が済まないという気持ちもうなずけますね。

さて，m_vset関数では音色の情報が入った配列の名前のほかに，それを何番目の音色にするかという値を引数とします。音色の番号は1から68まではプリセット音としてあらかじめ設定されていますから，通常は69～200を指定します（もちろん，1～68の音色を設定し直しても構いません)。それでは，m_vset関数のフォーマットを以下に示します。

m_vset(vo, va)

X-BASICの基礎事項(前回まで)

X-BASIC では，変数を使用する前に変数の型宣言をしなければなりません。宣言できるデータ型はint（4バイト整数)，char（1バイト整数)，str(文字列)，float(実数）の4種類です。

X-BASIC のプログラムの実行はその大部分が関数の呼び出しによって行われます。それ以外は制御構造です。型宣言と制御構造と関数，これがX-BASICの3大要素です。

X-BASIC には画面上のキャラクタをスムーズに移動させるためのスプライト機能が備わっています。これにより最大128個のキャラクタを同時に移動させることができます。この移動のとき，パターンの反転，色の変更なども可能です。また，バックグラウンドと呼ばれる画面が2面あり，ここでは最大64×64個並べたキャラクタを背景として利用できます。バックグラウンド面上では，画面上のすべてのキャラクタが同時に移動します。

また，X-BASICでは65536色同時発色を特徴とするX68000のグラフィック機能を扱うことができます。色数が65536色であるのはグラフィック画面（実画面）が512×512ドットの場合ですが，色数を256色，16色と減らすことによって，実画面を2画面，4画面と増やすことができます。さらに，色数を16色，実画面数を1画面に限れば1024×1024ドットという大画面を扱うこともできます。また，複数個の実画面は高速に切り換えることができますし，それぞれをスクロールさせることもできます。この機能をうまく使えば，アニメーションも簡単です。

また，グラフィック画面の特徴として半透明機能があります。これは，グラフィックの実画面同士あるいはグラフィック画面とテキスト画面（スプライト画面）を重ね合わせて表示する機能です。この重ね合わせは，最も優先順位の高いグラフィック画面が半透明になることで実現されます。しかし，残念ながら半透明機能はX-BASIC から直接扱うことができません。メモリ上にマッピングされているX68000のビデオコントローラの内部レジスタを直接書き換えることで扱うことができます。

X68000ではグラフィック画面のみならず，テキスト画面もビットマップ方式を採用しています。さらに，テキスト画面は16色のパレットやスクロール機能も備わっています。このため，テキスト画面もグラフィック画面と対等に扱うことができます。たとえば，グラフィック画面の退避領域としてテキスト画面を使用することができます。

また，X68000にはマウスが標準で付いてきます。そして，X-BASIC ではこのマウスを扱うための関数が用意されています。マウスを入力装置とすることで操作性のよいプログラムを書くことができます。

X68000のハードウェアでスプライト，グラフィックと並ぶ3大特徴のひとつがFM音源用です。

X68000は，FM音源用のLSIとしてOPM(YM2151）を内蔵し，8オクターブ，8重和音のステレオ演奏を行うことができます。そして，X-BASICにはOPMに音楽を演奏させるためのインタフェイスとしてMML（ミュージック・マクロ・ランゲージ）と呼ばれる言語が用意されています。このMMLは演奏の繰り返し指定を簡単に記述できるという特長があります。また，音楽の演奏は割り込みによってほかのプログラムの実行と同時に行われるため，このFM音源をゲームプログラムなどのBGMとして利用することも可能でしょう。

引数　vo　音色番号(1～200)
　　　va　２次元配列の名前
　　　　dim char va(4,10)
　　　　で宣言されるもの

2)　音色の取り出し

音色が設定できるからには，すでに設定されている音色の情報を取り出すための関数がなければ片手落ちでず。X-BASIC では音色情報を取り出すための m_vget 関数が用意されています。このm_vget関数は，m_vset関数と同形式の char 型２次元配列(の名前)を引数とし，もうひとつの引数である音色番号で指定される音色の情報を取り出して，配列に格納します。この関数はプリセット音などを取り出して，少しだけ変更する場合に便利です。m_vget関数のフォーマットを以下に示します。

m_vget(vo, va)
引数　vo　音色番号(1～200)
　　　va　２次元配列の名前
　　　　dim char va(4,10)
　　　　で宣言されるもの

それでは，ここでm_vget関数を使ったプログラムを作ってみましょう。音色の情報は m_vget 関数によって配列のなかに簡単に取り込むことができますが，それを変更するとなると配列の内容を調べる手間が必要です。このとき，配列の内容（音色の情報）が，

```
10000  dim char va(4,10) = {
10010  …,…,…,…………
10020  …,…,…,…………
10030  …,…,…,…………
10040  …,…,…,…………
10050  …,…,…,…………        }
```

といったプログラムの形（つまり，行番号のあとに音色情報の記述がくる形）で出力されると，配列の内容は一目瞭然で音色の変更も楽になります。これなら，必要な要素を変更したあと，カーソルを出力の最初の行に持っていき，リターンキーをバシバシと押していくことでその設定をプログラム内に取り込むことができます。このプログラムがリスト１です。リスト１のプログラムを実行すると音色の番号を聞いてきますから，好きな番号を入力するとその番号の音色がプログラムの形でディスプレイ上にプリントされます(図４)。

3)　音色を変更する演奏例

音色を変更する理由はプリセット音の音色が気に入らないからにほかなりません。FM 音源は万能ではなく，所詮は自然の楽器の音色を電気の力で真似ているものですから，本物とまったく同じ音色を作り出す

図１　音色情報その１

	0	1	2	3	4	5	6	7	8	9	10
0	フィードバック & アルゴリズム	スロット マスク	ウェーブ フォーム	シンクロ	スピード	PMD	AMD	PMS	AMS	LR PAN	ダミー (無意味)
1	OP1 AR	OP1 D1R	OP1 D2R	OP1 RR	OP1 D1L	OP1 TL	OP1 KS	OP1 MUL	OP1 DT1	OP1 DT2	OP1 AMS-EN
2	OP2 AR	OP2 D1R	OP2 D2R	OP2 RR	OP2 D1L	OP2 TL	OP2 KS	OP2 MUL	OP2 DT1	OP2 DT2	OP2 AMS-EN
3	OP3 AR	OP3 D1R	OP3 D2R	OP3 RR	OP3 D1L	OP3 TL	OP3 KS	OP3 MUL	OP3 DT1	OP3 DT2	OP3 AMS-EN
4	OP4 AR	OP4 D1R	OP4 D2R	OP4 RR	OP4 D1L	OP4 TL	OP4 KS	OP4 MUL	OP4 DT1	OP4 DT2	OP4 AMS-EN

図２　音色情報その２

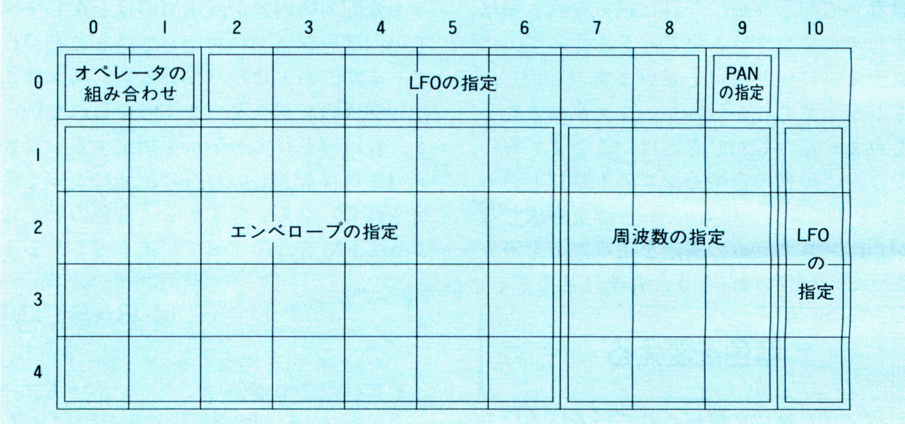


図３　音色情報その３

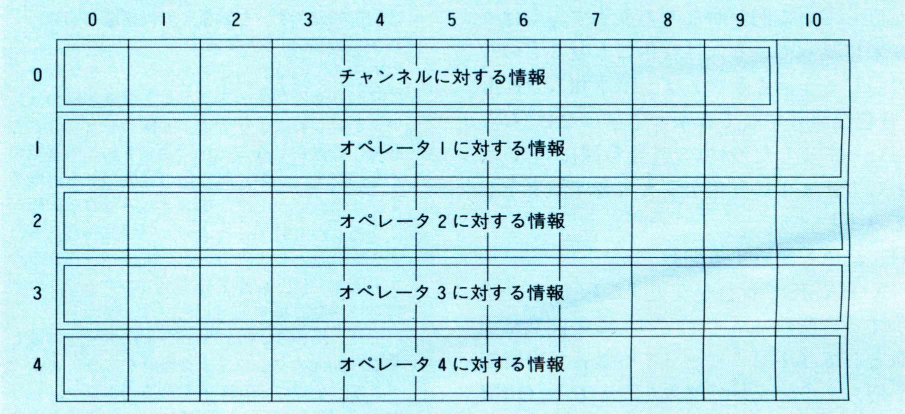


図４　リスト１の実行結果

```
10000 /*  音色( 1 )
10010 dim char va(4,10)={
10020 /* F/A MSK  WF SYC SPD PMD AMD PMS AMS PAN
10030 /*-----------------------------------------
10040      58, 15,  2,  0,220,  0,  0,  0,  0,  3,  0,
10050 /*  AR D1R D2R  RR D1L  TL  KS MUL DT1 DT2 AMS
10060 /*-----------------------------------------
10070      28,  4,  0,  5,  1, 37,  2,  1,  7,  0,  0, /* OP 1
10080      22,  9,  1,  2,  1, 47,  2, 12,  0,  0,  0, /* OP 2
10090      29,  4,  3,  6,  1, 37,  1,  3,  3,  0,  0, /* OP 3
10100      15,  7,  0,  5, 10,  0,  2,  1,  0,  0,  1} /* OP 4
```

のは不可能です。そこで，いろいろな人が「あーでもない」，「こーでもない」といって音色作りに精を出すわけです。しかしハッキリいって，音色を作ることは昨日今日の勉強でできるものではありません。音色に対してあまり慣れていない人の場合は，ほかの人たちが作った音色を聞き比べてみて，そのなかからピッタリとくるものを探すしかなさそうですね。

実際，「ピアノ（A.ピアノ）」の音色ひとつをとってみても，パソコンの機種別，音楽ソフト別に多くの種類があるようです(全部が異なる音色情報を持つ)。ここでは，Oh!X 3月号の67ページで紹介されたX1用VIPの「A.ピアノ」の3種類（もある）の音色と，X-BASICのプリセット音の「A.ピアノ」の音色（音色番号1）を聞き比べてみることにしましょう。比べて聞くと，それぞれ違う音色であっても，どれもそれなりに本物に聞こえるものですね。このためのプログラムがリスト2です。

リスト2ではX-BASICのプリセット音(リスト1のプログラムで取り出した)を音色番号70として再設定し，VIPによる音色を音色番号71～73で設定してあります。そして，音色番号を70から73まで変化させながら同じ曲を演奏させています。プログラム自身はfor～nextループのなかにMMLを書き並べてあるだけですので説明は不要でしょう。なお，この曲は，映画「機動戦士ガンダム　逆襲のシャア」の音楽より「メインタイトル・νガンダム」と呼ばれている曲の一部です。これでも，同じ「ピアノ」の音なのかなと思うくらい感じが違うのがわかります。そこで，結論。音色は演奏の感じを決定する大事な要素であるということです。

音楽の3要素を体験する

さて，音色にこだわることも大切なのですが，音を鳴らすだけでは音楽になりません。せっかく，MML を扱うからにはもっと音楽をやったという気分になれなくては面白くありませんね。これからしばらくは，MML を使って音楽の気分に浸ってみましょう。音楽の3要素といえば，リズム，メロディ，ハーモニーです。以下では，この音楽の3要素のそれぞれをMMLで実践します。

1）リズム

リズムとは，本来は「メロディの秩序だった動き」,「ハーモニーの秩序だった動き」を表す言葉なのだそうです。しかし，ここ

リスト1　音色の表示

```
 10 /*
 20 /*  音色の表示
 30 /*
 40 int i,j,lin=10000 : char v
 50 dim char va(4,10)
 60 input "音色の番号"; v : cls
 70 /*
 80 m_vget(v,va)       /*  音色データを取り出す
 90 /*
100 lpr() : print "/*  音色("; v; ")"
110 lpr() : print "dim char va(4,10)={"
120 /*
130 for i=0 to 4
140   if i=0 then {
150      lpr() : print "/* F/A MSK  WF SYC SPD PMD AMD PMS AMS PAN"
160      lpr() : print "/*-------------------------------------------"
170   } else if i=1 then {
180      lpr() : print "/*  AR D1R D2R  RR D1L  TL  KS MUL DT1 DT2 AMS"
190      lpr() : print "/*-------------------------------------------"
200   }
210   lpr() : print "   ";
220   for j=0 to 10
230      print using "###"; va(i,j);
240      if (i<>4)or(j<>10) then print ","; else print "}";
250   next
260   if i<>0 then print " /* OP";i else print
270 next
280 end
290 /*
300 /*  行番号のプリント
310 /*
320 func lpr()
330   print using "##### "; lin; : lin=lin+10
340 endfunc
```

リスト2　ピアノの音色の聞き比べ

```
 10 /*
 20 /*        逆襲のシャアより「メインタイトル (νGundam)」
 30 /*
 40 /*
 50 /*  A.ピアノ  プリセット
 60 /*
 70 dim char va0(4,10)={
 80 /*
 90 /* F/A MSK  WF SYC SPD PMD AMD PMS AMS PAN
100 /*-------------------------------------------
110      58, 15,  2,  0,220,  0,  0,  0,  0,  3,  0,
120 /*
130 /*  AR D1R D2R  RR D1L  TL  KS MUL DT1 DT2 AMS
140 /*-------------------------------------------
150      28,  4,  0,  5,  1, 37,  2,  1,  7,  0,  0, /* OP 1
160      22,  9,  1,  2,  1, 47,  2, 12,  0,  0,  0, /* OP 2
170      29,  4,  3,  6,  1, 37,  1,  3,  3,  0,  0, /* OP 3
180      15,  7,  0,  5, 10,  0,  2,  1,  0,  0,  1} /* OP 4
190 /*
200 m_vset(70,va0)
210 /*
220 /*
230 /*  A.ヒアノ  VIP
240 /*
250 dim char va1(4,10)={
260 /*
270 /* F/A MSK  WF SYC SPD PMD AMD PMS AMS PAN
280 /*-------------------------------------------
290      58, 15,  2,  1,220,  0,  4,  1,  1,  3,  0,
300 /*
310 /*  AR D1R D2R  RR D1L  TL  KS MUL DT1 DT2 AMS
320 /*-------------------------------------------
330      31,  5,  7,  4,  9, 37,  1,  1,  6,  0,  0, /* OP 1
340      22,  0,  4,  5,  4, 62,  1,  5,  2,  0,  0, /* OP 2
350      29,  0,  4,  5,  4, 77,  1,  1,  4,  0,  0, /* OP 3
360      31,  7,  6,  5,  4,  0,  2,  1,  1,  0,  1} /* OP 4
370 /*
380 m_vset(71,va1)
390 /*
400 dim char va2(4,10)={
410 /*
420 /* F/A MSK  WF SYC SPD PMD AMD PMS AMS PAN
430 /*-------------------------------------------
440      28, 15,  2,  0,180,  0,  1,  0,  1,  3,  0,
450 /*
460 /*  AR D1R D2R  RR D1L  TL  KS MUL DT1 DT2 AMS
470 /*-------------------------------------------
480      31, 20,  8, 10,  0, 24,  0,  1,  3,  0,  0, /* OP 1
490      31, 10,  5, 10,  0,  0,  0,  1,  4,  0,  1, /* OP 2
500      31, 20,  8, 10,  0, 45,  0,  3,  4,  0,  0, /* OP 3
510      25, 10,  5, 10,  0,  0,  3,  1,  3,  0,  1} /* OP 4
520 /*
530 m_vset(72,va2)
540 /*
550 dim char va3(4,10)={
560 /*
570 /* F/A MSK  WF SYC SPD PMD AMD PMS AMS PAN
580 /*-------------------------------------------
590      58, 15,  2,  0,205,  0,  0,  0,  0,  3,  0,
600 /*
610 /*  AR D1R D2R  RR D1L  TL  KS MUL DT1 DT2 AMS
620 /*-------------------------------------------
630      19,  2,  1,  4,  3, 33,  3,  5,  7,  0,  0, /* OP 1
640      19,  2,  1,  4,  3, 25,  3,  5,  2,  0,  0, /* OP 2
```

では単純に音楽を演奏するうえでの拍子の取り方と思ってください。要するに「タン・タッ・タ」,「タン・タ・タン」,「タン・タ・タ・タン」とかいう,一定のパターンを持ったビートのことです。「ワルツ」だとか「タンゴ」,「サンバ」といった言葉を聞いたことがあるでしょう。それらは皆,ある特定のパターンを持ったリズムのことなのです。図5に代表的なリズムパターンを示します。それでは,図5に示されたリズムパターンをMMLで演奏させてみましょう。そのためのプログラムがリスト3です。このプログラムも単純ですから説明は不要でしょう。

なお,このプログラムでは,演奏する音符の列がまったく同じため,4ビート(4拍子)のリズムは2ビート(2拍子)のリズムと同じもの,6ビート(6拍子)のリズムは3ビート(3拍子)のリズムと同じものとして省略してあります。しかし,本当は強拍とか弱拍とかいった区別をしなければならないところです。たとえば,

2拍子 → 強―弱
3拍子 → 強―弱―弱
4拍子 → 強―弱―中強―弱
6拍子 → 強―弱―弱―中強―弱―弱

などといった強弱をつける必要があります(こんなの小学校で習った記憶があるなあ)。具体的には音量(Vコマンド)を変えて強弱をつければよいので,暇な人はリスト3を書き換えてみてください。

2) ハーモニー
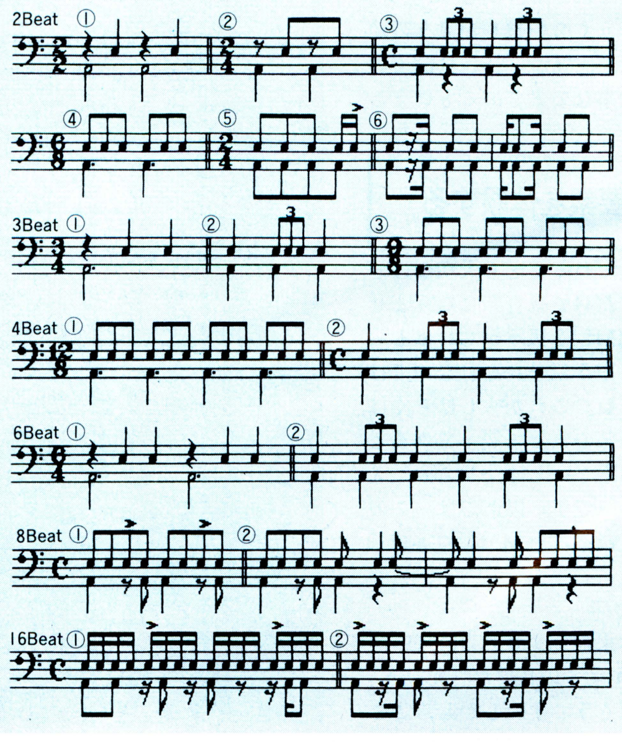
ハーモニーとは,メロディを包み込んで曲に調和感を与えるコード(和音)のつながりのことです。要するに,いくつかの音(基本的には3つ)を同時に鳴らすことです。楽譜のなかにCとかAmとかBdimとか書かれているのをよく見かけますが,それがコードの指定です。CとかAとかBという名前はコードを形成するいちばん低い音の名前で,mとかdimというのは3つの音の音程の関係を示します。mとかdimの意味は次のとおりです。なお,「全」,「半」は,それぞれ,全音,半音の関係を表しています。

・なにもなし(メジャー)
　♩♩－全＋全－♩♩－全＋半－♩♩
・m(マイナー)
　♩♩－全＋半－♩♩－全＋全－♩♩
・dim(ディミニッシュ)
　♩♩－全＋半－♩♩－全＋半－♩♩
・aug(オーギュメント)
　♩♩－全＋全－♩♩－全＋全－♩♩

コードにこういった区別があるのは,コ

```
650      19,  2,  1,  4,  3, 31,  2,  1,  4,  0,  0, /* OP 3
660      19,  2,  1,  4,  3,  0,  3,  1,  7,  0,  1} /* OP 4
670 /*
680 m_vset(73,va3)
690 /*
700 for i=70 to 73 : print "この演奏は音色";i;"です"
710 /*
720 m_init() : for j=1 to 5 : m_alloc(j,2000) : next
730 /*
740 m_tempo(160)
750 /*
760 m_trk(1,"V15 L8") : m_trk(1,"@"+str$(i))
770 m_trk(2,"V8  L8") : m_trk(2,"@"+str$(i))
780 m_trk(3,"V8  L8") : m_trk(3,"@"+str$(i))
790 m_trk(4,"V8  L8") : m_trk(4,"@"+str$(i))
800 m_trk(5,"V8  L8") : m_trk(5,"@"+str$(i))
810 m_trk(1,"O4 A4 |: O5 D2.{ERF}4 E2&>A8R8A4 <D2C4>{GR<C}4 >A2. R4 ")
820 m_trk(2,"   R4 |: O4 DDDDDDDD  EEEEEEEE   DDDDEEEE       FFFFFFFF")
830 m_trk(3,"   R4 |: O3 AAAAAAAA <CCCCCCCC  >B+B+B+B+<CCCC CCCCCCCC")
840 m_trk(4,"   R4 |: O3 FFFFFFFF  AAAAAAAA   FFFFGGGG       AAAAAAAA")
850 m_trk(5,"   R4 |: O3 RRRRRRRR  GGGGGGGG   RRRRRRRR       RRRRRRRR")
860 m_trk(1,"|1 O4 G2A4{ERG}4 F2&FA<DE F4F4G4.F E2.>A4          :|")
870 m_trk(2,"|1 O4 EEEEEEEE   DDDDDDDD DDDDDDDD DDDDC+C+C+C+    :|")
880 m_trk(3,"|1 O4 C+C+C+C+C+C+C+C+ >AAAAAAAA BBBBBBBB AAAAAAAA :|")
890 m_trk(4,"|1 O3 AAAAAAAA   FFFFFFFF G+G+G+G+G+G+G+G+GGGGGGGG :|")
900 m_trk(5,"|1 O3 GGGGGGGG   RRRRRRRR FFFFFFFF RRRRRRRR        :|")
910 m_trk(1,"|2 O4 G2A4R8<E8 F2&F>A<DE F4F4G4.F E2.E4            ")
920 m_trk(2,"|2 O4 EEEEEEEE   DDDDDDDD DDDDDDDD DDDDC+C+C+C+      ")
930 m_trk(3,"|2 O4 C+C+C+C+C+C+C+C+ >AAAAAAAA BBBBBBBB AAAAAAAA   ")
940 m_trk(4,"|2 O3 AAAAAAAA   FFFFFFFF G+G+G+G+G+G+G+G+GGGGGGGG   ")
950 m_trk(5,"|2 O3 GGGGGGGG   RRRRRRRR FFFFFFFF RRRRRRRR          ")
960 m_trk(1,"O5 F2EDC>B- A2.A4 G4AB-A4G4 F2&FFAF <D4>{FFF}4A<D4E FRG2F4")
970 m_trk(2,"O4 DDDDEEEE EEEEEEEE FFFFEEEE FFFFFFFF FFFFFFFF FFFFFFFF  ")
980 m_trk(3,"O3 B-B-B-B-<CCCCCCCCCCCCDDDDC+C+C+C+>AAAAAAAABBBBBBBBBBBBBBBB")
990 m_trk(4,"O3 GGGGGGGGAAAAAAAAB-B-B-B-AAAAFFFFFFFFAAAAAAAAG+G+G+G+G+G+G+G+")
1000 m_trk(5,"O3 RRRRB-B-B-B-FFFFFFFFGGGGGGGG RRRRRRRR FFFFFFFF FFFFFFFF")
1010 m_trk(1,"O5 E2.>A4  <F2E4F&E D2.D4 B-2A&GF&G A2C2")
1020 m_trk(2,"O4  R4E.D16C+2 R>A<D>ARA<D>A RFB-FRFBF RDFDREB-E RCECRCEC")
1030 m_trk(3,"O3  GA&A2.     D2C2         >B-2A2        G2<C2      >F2F2  ")
1040 m_trk(4,"O3  R1         R1             R1          R1         R1   ")
1050 m_trk(5,"O3  R1         R1             R1          R1         R1   ")
1060 m_trk(1,"O5 G2F&ED&E F2.F4 E4.EE&DC+&D E2.>A4")
1070 m_trk(2,"O2 RB-<D>B-R<EGE RAB<C+DEF4 >RDFARDEG+ REDEA2")
1080 m_trk(3,"O2 E2A2          <D1            B2B2       A1    ")
1090 m_trk(4,"O3 R1             R1             R1         R1    ")
1100 m_trk(5,"O3 R1             R1             R1         R1    ")
1110 m_trk(1,"O5 D2.{EEF}4 E2.>A4 <D2C4>{GG<C}4 >A2.A4 G2.{AR<E}4 F2G2 A2.R4")
1120 m_trk(2,"O4 DDDDDDDD EEEEEEEE DDDDEEEE EEEEEEEE DDDDC+C+C+C+ DDDDDDDD D2.R
4")
1130 m_trk(3,"O3 AAAAAAAA<CCCCCCCC>B-B-B-B-<CCCCCCCCCCCCB-B-B-B-C+C+C+C+AAAAB-B
-B-B- A2.R4")
1140 m_trk(4,"O3 FFFFFFFF AAAAAAAA FFFFGGGG AAAAAAAA GGGGGGGG FFFFGGGG G2.R4")
1150 m_trk(5,"O3 R1        GGGGGGGG R1        FFFFFFFF EEEEEEEE R1        E2.R4")
1160 /*
1170 m_play()
1180 /*
1190 while m_stat() : endwhile
1200 next
```


図5　リズムパターン



ードが音階（キー）を元にして作られているからです。すなわち，コードとは音階のなかで3つの音をひとつおき（3度の音程）に拾ったものにほかなりません。図6に長音階（ハ長調）と短音階（イ短調）の例を示します。これらの音階の上にコードを作る（ひとつおきに3つの音を拾う）と図7に示すような名前のコードができるのです（コードの命名は意外と安直なのですよ）。

なお，図6で示す短音階にはGの音に♯が付いています。これは短音階のうち，和声的短音階と呼ばれる音階です（ほかに自然的短音階，旋律的短音階がありますが，ここでは省略ね）。紙に書かれたコードだけを見ていても実感がわきませんから，MMLで演奏してみましょう（楽譜を見ただけで実感できる人がうらやましいなあ）。そのプログラムがリスト4（長音階）とリスト5（短音階）です。8重和音を特長とするOPMでは，たかが3音の和音なんてたやすいことですね。

さて，音楽の演奏ではメロディが移り変わるのに従って，コードも移り変わっていきます。このコード進行には，クラシック

リスト3　リズムパターン演奏プログラム

```
 10 /*
 20 /*  リズム
 30 /*
 40 int i
 50 dim str rhy0(13)[50],rhy1(15)[50],rhy2(15)[50]
 60 rhy0(0)=" 2 Beat 1"
 70 rhy1(0)="O3 R4E4R4E4"
 80 rhy2(0)="O2 A2  A2  "
 90 rhy0(1)=" 2 Beat 2"
100 rhy1(1)="O3 R8E8R8E8"
110 rhy2(1)="O2 A4  A4  "
120 rhy0(2)=" 2 Beat 3 (Jazz)"
130 rhy1(2)="O3 C4{CCC}4C4{CCC}4"
140 rhy2(2)="O2 F4R4    F4R4    "
150 rhy0(3)=" 2 Beat 4 //  4 Beat 1"
160 rhy1(3)="O3 C8C8C8C8C8C8"
170 rhy2(3)="O2 F4.   F4.   "
180 rhy0(4)=" 2 Beat 5 (Tango)"
190 rhy1(4)="O3 V8 F8F8F8F16 V10 F16"
200 rhy2(4)="O2    A8A8A8A8         "
210 rhy0(5)=" 2 Beat 6 (Habanera)"
220 rhy1(5)="O3 E8R16E16E8E8 E16E8E16E8E8"
230 rhy2(5)="O2 A8R16A16A8A8 A16A8A16A8A8"
240 rhy0(6)=" 3 Beat 1 (Waltz) //  6 Beat 1"
250 rhy1(6)="O3 R4E4E4"
260 rhy2(6)="O2 A2.   "
270 rhy0(7)=" 3 Beat 2 (Jazz)  //  6 Beat 2"
280 rhy1(7)="O3 E4{EEE}4E4"
290 rhy2(7)="O2 A4 A4   A4"
300 rhy0(8)=" 3 Beat 3"
310 rhy1(8)="O3 E8E8E8E8E8E8E8E8E8"
320 rhy2(8)="O2 A4.   A4.   A4.
330 rhy0(9)=" 4 Beat 2 (Jazz)"
340 rhy1(9)="O3 E4{EEE}4E4{EEE}4"
350 rhy2(9)="O2 A4A4    A4A4    "
360 rhy0(10)=" 8 Beat 1 (Rock)"
370 rhy1(10)="O3 V8E8E8V10E8V8E8 E8E8V10E8V8E8"
380 rhy2(10)="O2   A4      R8  A8 A4      R8  A8"
390 rhy0(11)=" 8 Beat 2 (Latin)"
400 rhy1(11)="O3 E8E8E8E8E8E4E8 & E8E4E8E8E8E8E8"
410 rhy2(11)="O2 A4  R8A8A4R4      A4R8A8A4  R4  "
420 rhy0(12)="16 Beat 1 (Rock)"
430 rhy1(12)="O3 L16 V8EEEE V10EV8EEE EEEE V10EV8EEE"
440 rhy2(12)="O2 L8    A A  R16  AR16 A R  R16  A16A"
450 rhy0(13)="16 Beat 2 (Latin)"
460 rhy1(13)="O3 L16 V10EV8EEE V10EV8EEE V10EV8EEE V10EV8EEE"
470 rhy2(13)="O2 L8    AR16A16    A   R    AR16A16    A  R  "
480 /*
490 for i=0 to 13
500     print rhy0(i)
510     m_init()
520     m_alloc(1,300)  : m_alloc(2,300)
530     m_assign(1,1)   : m_assign(2,2)
540     m_tempo(120)
550     m_trk(1,"V8@59"): m_trk(2,"V8@48")
560     while (m_free(1)>100) and (m_free(2)>100)
570         m_trk(1,rhy1(i)) : m_trk(2,rhy2(i))
580     endwhile
590     m_play()
600     while m_stat() : endwhile
610 next
```

図6　長音階と短音階

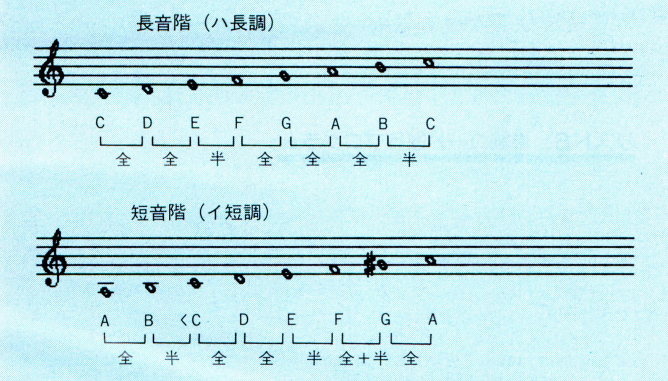


図7　メジャーコードとマイナーコード

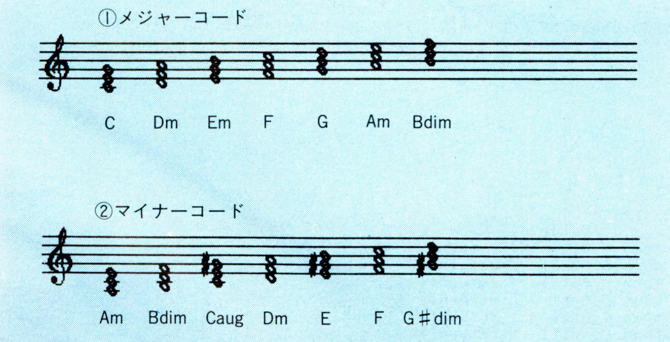


図8　古典的コード進行

C（ハ長調）

初めのコード	次のコードの順位 第1級	第2級	第3級	第4級
C	F　G	Am　Em	Dm	
Dm	G	Am	C　Em	F
Em	F	Am　G	Dm	C
F	C　G	Dm　Em		Am
G	C　Am	Em		Dm　F
Am	F　Dm	G　Em		C

Am（イ短調）

初めのコード	次のコードの順位 第1級	第2級	第3級	第4級
Am	Dm　E	F　Bdim		
Bdim	E			
Dm	Bdim　Am	E		
E	Am　F			F
F	Dm　E	Bdim　Am		Dm

音楽などの時代から伝わる一般的な規則があって，現代の音楽にもかなりの影響を与えているようです。図8がハ長調とト短調における古典的なコード進行ですが，このコード進行をMMLで演奏してみます。図でわかるように，あるコードに対して次のコードは一意には定まりません。そこで，乱数を用いてコードを決定することにします。

簡単にするため，ここでは第1級と第2級のコードのみを使った進行を行いましょう。もちろん，第1級のコードが選ばれる確率は第2級のコードよりも高いものとし，同じ級のコードは同じ確率で選ばれるものとします。このような方針で作ったプログラムがリスト6です。プログラムのアルゴリズムは次のようになっています。

- チャンネル1～3を使用してコードを演奏する。チャンネル番号はトラックの番号に等しくする。
- 最初のコードは，長調ならばC，短調ならばAm。
- あるコードの次のコードは，長調ならばnxt_cod_maj関数により，短調ならばnxt_cod_min関数によって求める。求め方は図8の表に従うものとする。
- トラックバッファの容量がなくなるか20小節分のコードを設定するまで，コードの設定を繰り返す。
- トラックバッファへの設定が終わったら演奏する。

この程度の説明でプログラムはわかると思います。まあ，あれこれ考えるより，実際に入力してみましょう。実際に演奏してみると，さすが古くから生き残ってきたコード進行だけあって，心地よい響きを持っていますね。

3) メロディ

メロディとは，高さや長さの異なる音の連続的な進行です。メロディは曲そのものを決定づけてしまいますから，代表的なメロディというものはありません（あったら盗作だ)。まあ，音の流れ方（連続する数個の音の関係，始まりと終わりの音）についての法則があるくらいでしょうか。既存の曲のメロディをそのままMMLで演奏しても楽しくない（えっ，そっちのほうがいいって？）ので，ここではコード進行の例と同様に乱数を用いてメロディを作り出してみましょう。ただ，本当に乱数だけを用いたのでは，ほとんど意味不明のメロディができてしまいますから，音の生成にある程度の規則をつけます。つまり，次のような規則です。

リスト4　長音階コード演奏プログラム

```
 10 /*
 20 /*  コードの練習（長調）
 30 /*
 40 m_init()
 50 /*
 60 m_alloc(1,800) : m_alloc(2,800) : m_alloc(3,800)
 70 /*
 80 m_assign(1,1)  : m_assign(2,2)  : m_assign(3,3)
 90 /*
100 m_tempo(120)
110 /* C
120   m_trk(1,"cccc") : m_trk(2,"eeee") : m_trk(3,"gggg")
130 /* Dm
140   m_trk(1,"dddd") : m_trk(2,"ffff") : m_trk(3,"aaaa")
150 /* Em
160   m_trk(1,"eeee") : m_trk(2,"gggg") : m_trk(3,"bbbb")
170 /* F
180   m_trk(1,"ffff") : m_trk(2,"aaaa") : m_trk(3,"<cccc>")
190 /* G
200   m_trk(1,"gggg") : m_trk(2,"bbbb") : m_trk(3,"<dddd>")
210 /* Am
220   m_trk(1,"aaaa") : m_trk(2,"<cccc>") : m_trk(3,"<eeee>")
230 /* Bdim
240   m_trk(1,"bbbb") : m_trk(2,"<dddd>") : m_trk(3,"<ffff>")
250 /*
260 m_play(1,2,3)
```

リスト5　短音階コード演奏プログラム

```
 10 /*
 20 /*  コードの練習（短調）
 30 /*
 40 m_init()
 50 /*
 60 m_alloc(1,800) : m_alloc(2,800) : m_alloc(3,800)
 70 /*
 80 m_assign(1,1)  : m_assign(2,2)  : m_assign(3,3)
 90 /*
100 m_tempo(120)
110 /*
120 /* Am
130   m_trk(1,">aaaa<") : m_trk(2,"cccc") : m_trk(3,"eeee")
140 /* Bdim
150   m_trk(1,">bbbb<") : m_trk(2,"dddd") : m_trk(3,"ffff")
160 /* Caug
170   m_trk(1,"cccc") : m_trk(2,"eeee") : m_trk(3,"g+g+g+g+")
180 /* Dm
190   m_trk(1,"dddd") : m_trk(2,"ffff") : m_trk(3,"aaaa")
200 /* E
210   m_trk(1,"eeee") : m_trk(2,"g+g+g+g+") : m_trk(3,"bbbb")
220 /* F
230   m_trk(1,"ffff") : m_trk(2,"aaaa") : m_trk(3,"<cccc>")
240 /* G#dim
250   m_trk(1,"g+g+g+g+") : m_trk(2,"bbbb") : m_trk(3,"<dddd>")
260 /*
270 m_play(1,2,3)
```

リスト6　単純コード進行プログラム

```
 10 /*
 20 /*  ランダムコード進行
 30 /*
 40 int s=0
 50 int chosi=1      /* 0 は長調  1 は短調
 60 str chord="C"
 70 input "0->長調  1->短調  "; chosi
 80 if chosi=1 then chord="Am"
 90 rnd_init()
100 m_init()
110 m_alloc(1,800) : m_alloc(2,800) : m_alloc(3,800)
120 m_assign(1,1)  : m_assign(2,2)  : m_assign(3,3)
130 m_tempo(120)
140 m_trk(1,"V15 @1  P3 L4")
150 m_trk(2,"V15 @1  P3 L4")
160 m_trk(3,"V15 @1  P3 L4")
170 /*
180 while nokori(3) and (s<20)
190   C_set(1,chord)
200   if chosi=0 then chord=nxt_cod_maj(chord) else chord=nxt_cod_min(chord)
210   if (s mod 4)=3 then print
220   s=s+1
230 endwhile
240 /*
250 m_play(1,2,3)
260 /*
270 end
280 /*
290 /* コードを設定する
300 /*
310 func C_set(t,c;str)
320 color 2: print c,; : color 3
330   if c="C"    then C_C(t)   : return()
340   if c="Dm"   then C_Dm(t)  : return()
350   if c="Em"   then C_Em(t)  : return()
360   if c="F"    then C_F(t)   : return()
370   if c="G"    then C_G(t)   : return()
380   if c="Am"   then C_Am(t)  : return()
390   if c="Bdim" then C_Bdim(t): return()
400   C_E(t)
```

・音階は長音階のみ。
・音の長さは全音符，２分音符，４分音符，８分音符のうちのどれかで，出現する確率は，

全＜２分＜８分＜４分

の順とする。
・最初の音はド(C)，ミ(E)，ソ(G)のうちのどれかで，使用する確率は，

ド＜ミ＜ソ

の順とする。
・連続する２つの音の関係は１度から６度の間とし，出現する確率は，

５度＜６度＜１度＜４度＜２度＜３度

の順とする。

以上のような４つの規則に従ってメロディを作るプログラムがリスト７です。プログラムの説明を簡単にしましょう。音の選び方としては長調（ハ長調）を仮定していますから，

C，D，E，F，G，A，B

のみを用います。高さの度数の計算もこれらの音の間で計算します。これが基本です。実際の動作は，まず，最初の音（名）をsaisho関数で，音長をnagasa関数で決定します。そのあとはtakasa関数によって音名を，nagasa関数によって音長を求め，それをnnn関数によってMMLに変換しながらトラックバッファ１に入れていき，これをトラックバッファが空になるまで（正確には，残り容量が30バイト以下になるまで）繰り返します。そして，最後に演奏を始めます。要はこれだけのプログラムで，くどくど説明してもしようがないのですが，音名(takasa関数による)を求める関数は多少複雑なのでもう少し詳しく説明します。先に述べたように，音名はひとつ前の音に対して，高さの差が，

５度＜６度＜１度＜４度＜２度＜３度

の確率で現れるように求めます。しかし，高さの差を決めただけでは，それが前の音より高い音なのか低い音なのかが決まりません。そこで，ここではオクターブ４に向かう方向に変化するようにします。つまり，ひとつ前の音のオクターブ(okuという変数)を覚えておいて，もし，それが４より高いオクターブであれば，次の音は前の音よりも低い音にします。一方，ひとつ前の音のオクターブが４より低ければ，次の音は前の音より高い音にします。リスト７では，常にそうしているわけではなく，確率的にそういう傾向を持たせているので複雑になっているのです。

また，オクターブ０と８においては使える音が限られるため，ここでオクターブ０

```
410 endfunc
420 /* C
430 func C_C(t)
440   m_trk(t,"cccc") : m_trk(t+1,"eeee") : m_trk(t+2,"gggg")
450 endfunc
460 /* Dm
470 func C_Dm(t)
480   m_trk(t,"dddd") : m_trk(t+1,"ffff") : m_trk(t+2,"aaaa")
490 endfunc
500 /* Em
510 func C_Em(t)
520   m_trk(t,"eeee") : m_trk(t+1,"gggg") : m_trk(t+2,"bbbb")
530 endfunc
540 /* F
550 func C_F(t)
560   m_trk(t,"ffff") : m_trk(t+1,"aaaa") : m_trk(t+2,"<cccc>")
570 endfunc
580 /* G
590 func C_G(t)
600   m_trk(t,"gggg") : m_trk(t+1,"bbbb") : m_trk(t+2,"<dddd>")
610 endfunc
620 /* Am
630 func C_Am(t)
640   m_trk(t,">aaaa<") : m_trk(t+1,"cccc") : m_trk(t+2,"eeee")
650 endfunc
660 /* Bdim
670 func C_Bdim(t)
680   m_trk(t,">bbbb<") : m_trk(t+1,"dddd") : m_trk(t+2,"ffff")
690 endfunc
700 /* E
710 func C_E(t)
720   m_trk(t,"eeee") : m_trk(t+1,"g+g+g+g+") : m_trk(t+2,"bbbb")
730 endfunc
740 /*
750 /* つぎのコード(長調)
760 /*
770 func str nxt_cod_maj(c;str)
780   if c="C" then {
790           if ransu(2)=1 then return( either("F","G") )
800           return( either("Am","Em") )
810   } else if c="Dm" then {
820           if ransu(2)=1 then return( "G" )
830           return( "Am" )
840   } else if c="Em" then {
850           if ransu(2)=1 then return( "F" )
860           return( either("Am","G") )
870   } else if c="F"  then {
880           if ransu(2)=1 then return( either("C","G") )
890           return( either("Dm","Em") )
900   } else if c="G"  then {
910           if ransu(2)=1 then return( either("C","Am") )
920           return( "Em" )
930   } else { /* Am
940           if ransu(2)=1 then return( either("F","Dm") )
950           return( either("G","Em") )
960   }
970 endfunc
980 /*
990 /* つぎのコード(短調)
1000 /*
1010 func str nxt_cod_min(c;str)
1020   if c="Am" then {
1030           if ransu(2)=1 then return( either("Dm","E") )
1040           return( either("F","Bdim") )
1050   } else if c="Bdim" then {
1060           return( "E" )
1070   } else if c="Dm"   then {
1080           if ransu(2)=1 then return( either("Bdim","Am") )
1090           return( "E" )
1100   } else if c="E"    then {
1110           return( either("Am","F") )
1120   } else { /* F
1130           if ransu(2)=1 then return( either("Dm","E") )
1140           return( either("Bdim","Am") )
1150   }
1160 endfunc
1170 /*
1180 /* どちらかの文字列を選択
1190 /*
1200 func str either(a;str,b;str)
1210   if int(rnd()*2) then return(a) else return(b)
1220 endfunc
1230 /*
1240 /*  乱数の初期化
1250 /*
1260 func rnd_init()
1270   str t : int h,m,s
1280   t=time$
1290   h=val(left$(t,2)) : m=val(mid$(t,4,2)) : s=val(mid$(t,7,2))
1300   randomize((h*m*s)and &H7FFF)
1310 endfunc
1320 /*
1330 /*  残りトラック
1340 /*
1350 func int nokori(n)
1360   int p=-1,i
1370   for i=1 to n : p=p and (m_free(i)>50) :next
1380   return(p)
1390 endfunc
1400 /*
1410 /*  乱数(0に近い値ほど出にくい)
1420 /*
1430 func int ransu(n)
1440   return( int(sqr(rnd())*n) )
1450 endfunc
```

をオクターブ1に，オクターブ8をオクターブ7に無理矢理変更するという手抜きをやっています。しかし，オクターブは4に向かうようにしてあって，オクターブ0とかオクターブ8になることが確率的に小さいので問題はないでしょう。それなら，最初から小細工をするなといわれそうですが，たまたまオクターブの0や8になったときエラーで止まるのがいやだっただけです。

それでは，リスト7のプログラムによって作られたメロディの冒頭部分を図9に示しましょう。

自動作曲プログラム

これまで，リズム，ハーモニー，メロディと，音楽の構成要素を細切れに見てきましたが，ここでは，それらを組み合わせることを考えましょう。つまり，作曲をするプログラムです。確かに，音楽を作曲するには才能が必要です。しかし，こちらには乱数という強い味方がついています。この乱数によって，いろいろな曲を次々と無限に作り出し，売れっ子作曲家の気分を味わってみることにしましょう（でも，この場合の作曲者は「乱数」ということになるのかな）。

1） アルゴリズム

この作曲プログラムの原型はリスト6のコード進行プログラムです。まず，乱数によってコードの進行を決めます。このとき，コードの演奏はリスト6と同じく



というリズムで演奏します。少し単調ですが，まあ，典型的な4分の4拍子ですね。そして，このコード上にメロディを載せるのです。メロディはコードを構成する3つの音を基本として作りますが，コードを作る音しか出現しない曲というのはまれですから，ときどきコード以外の音も混ぜてやることにしましょう。

メロディの決め方は，基本的には，リスト7のメロディ作成プログラムと同様に，ひとつ前の音に対する高さの差を乱数で決めて次の音を求めることにします。このとき，求めた音がコードを作る音でないときは，確率的にコードを作るいちばん近い音に補正してやります。確率的というのは，ごくまれにはコードを作らない音であっても，最初に求めた音をそのまま使うということです。さて，一度決めたコードは1小節にわたって有効とすることにしましょう。

このとき，そのコードに対する1小節分

リスト7　メロディ作成プログラム

```
10 /* でたらめ自動作曲
20 /*
30 dim str nn(11)={"C","C+","D","D+","E","F","F+","G","G+","A","A+","B"}
40 dim char onkai(6)={0,2,4,5,7,9,11}
50 int l,n,oku=4,old_oku=4
60 rnd_init()
70 /*
80 m_init() : m_alloc(1,1000) : m_assign(1,1) : m_tempo(120)
90 m_trk(1,"V10@1")
100 /*
110 l=nagasa() : n=saisho() : m_trk(1,nnn(l,n))
120 /*
130 while m_free(1)>30
140        l=nagasa() : n=takasa(n) : m_trk(1,nnn(l,n))
150 endwhile
160 /*
170 m_play(1)
180 /*
190 end
200 /*
210 /*  乱数の初期化
220 /*
230 func rnd_init()
240   str t : int h,m,s
250   t=time$
260   h=val(left$(t,2))
270   m=val(mid$(t,4,2))
280   s=val(mid$(t,7,2))
290   randomize((h*m*s) and &H7FFF)
300 endfunc
310 /*
320 /*  乱数
330 /*
340 func int ransu(n)
350   return( int(sqr(rnd())*n) )
360 endfunc
370 /*
380 /*  乱数 （その2）
390 /*
400 func int ransu2(n)
410   return( int(sqr(sqr(rnd()))*n) )
420 endfunc
430 /*
440 /* 音の高さと長さをMMLに変換
450 /*
460 func str nnn(l,n)
470    str r
480     r=nn(onkai(n))+str$(l)
490     if oku<>old_oku then r="O"+str$(oku)+r
500     old_oku=oku
510     print chr$(5);r ;" ";
520     return(r)
530 endfunc
540 /*
550 /*  最初の音
560 /*
570 func int saisho()
580     switch ransu(3)
590     case 0: return(0)
600     case 1: return(2)
610     default:return(4)
620     endswitch
630 endfunc
640 /*
650 /* 長さ 4 > 8 > 2 > 1 の順
660 /*
670 func int nagasa()
680   switch ransu(4)
690   case 0: return(2)
700   case 1: return(4)
710   default:return(8)
720   endswitch
730 endfunc
740 /*
750 /* 高さ 差が 3 > 2 > 4 > 1 > 6 > 5  の順
760 /*
770 func int takasa(mae)
780   int i
790   switch ransu(6)
800   case 0: i=4 : break
810   case 1: i=5 : break
820   case 2: i=0 : break
830   case 3: i=3 : break
840   case 4: i=1 :break
850   default:i=2 : break
860   endswitch
870   if oku>4 then {
880      if ransu2(2) then i= -i
890   } else if oku<4 then {
900      if ransu2(2)=0 then i= -i
910   } else {
920      if int(rnd()*2) then i= -i
930   }
940   i=i+mae
950   if i<0 then {
960      oku=oku-1 : i=i+7
970   }else if i>6  then{
980      oku=oku+1 : i=i-7
990   }
1000  if oku=0 then oku=1
1010  if oku=8 then oku=7
1020  return( i )
1030 endfunc
1040 /*
```

のメロディを作らなければなりません。ひとつの音の長さはリスト7と同様に決めますが，ここではその長さを覚えておいて，1小節分のメロディができるまで

・ひとつ前の音から，乱数によって次の音を作り出す（大部分は指定したコードを構成する音になるようにする)。
・乱数によって音の高さを決める（オクターブの処理はリスト7と同じ)。

という操作を繰り返すことにします。1小節分の長さを計算するためには，得られる音の長さを

全音符	→	8
2分音符	→	4
4分音符	→	2
8分音符	→	1

として（これ以外の長さは出現しないようにしてある)，合計が8になるまで先の操作を繰り返します（わかりますね)。しかし，このままだと，たとえば，長さに対応する数値の合計が7のところで，次の音符が2分音符（数値4）だった場合など，合計が7＋4＝11で8（1小節分）を越えてしまいます。こういう場合は次の音を加えず休符を1小節の長さになるまで埋めることに

乱数に関する基礎知識

X-BASIC では乱数を発生する関数として，rnd関数とrand関数が用意されています。このうち，rnd関数は0以上1未満の乱数を発生し，rand関数は0以上2^{15}未満の乱数を発生します。明らかに2つの乱数関数の関係は

rand( )≡int (rnd( )×2^{15})

となっています。また，0以上n未満の乱数を発生する関数は

int(rnd( )×n)

という式で作り出すことができます。

ところで，rnd関数やrand関数は一様乱数と呼ばれる乱数，つまり，すべての数値の発生確率が同じ乱数を発生します。このため，今回のように，ある数値と別の数値の発生する確率が異なる乱数を発生させる必要がある場合は，少々細工をしなければなりません。その一例として平方根があります。

sqr(rnd( ))

はrnd関数と同じく，0以上1未満の乱数を発生します。このとき，たとえば，0.5未満の乱数を発生させるためには，rnd関数の値は0.25未満でなくてはなりません。つまり，rnd関数の平方根でできる関数は，rnd関数に比べて0.5未満の乱数を発生する確立が小さくなっています。同様に考えていくと，これは0に近い値ほど出にくくなる関数であることがわかります。そこで，0以上n未満の乱数を発生する関数で，0に近い値ほど出にくくなる関数は

int(sqr(rnd( ))×n)

という式に従って作ればよいことがわかります。さらに，
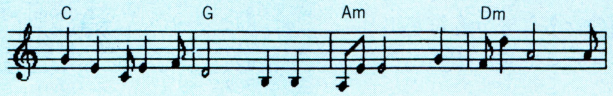
int(sqr(sqr(rnd( )))×n)

では，0に近い値がさらに出にくくなります。

図9　乱数によるメロディ（冒頭）



図10　自動作曲（冒頭）



リスト8　自動作曲プログラム

```
 10 /*
 20 /* 「これで私も作曲家」プログラム
 30 /*
 40 dim str nn(11)={"C","C+","D","D+","E","F","F+","G","G+","A","A+","B"}
 50 dim char onkai(6)={0,2,4,5,7,9,11}
 60 int oku=4,old_oku=4,sum_nag,tmp_nag,i,old_n,s=0,maxtr=8,owari
 70 dim int save_n(8)
 80 dim int save_oku(8)={4,4,4,4,4,4,4,4,4}
 90 int chosi=0       /* 0 は長調  1 は短調
100 str chord="C",tmp_str
110 input "0->長調  1->短調 "; chosi
120 if chosi=1 then chord="Am"
130 rnd_init()
140 m_init()
150 for i=1 to 8      : m_alloc(i,800) : next
160 for i=1 to maxtr : m_assign(i,i)  : next
170 m_tempo(120)
180 m_trk(1,"V10 @1  P3 L4"): m_trk(2,"V10 @1  P3 L4")
190 m_trk(3,"V10 @1  P3 L4"): m_trk(4,"V15 @1  P3 L4")
200 m_trk(5,"V15 @1  P3 L4"): m_trk(6,"V15 @1  P3 L4")
210 m_trk(7,"V15 @1  P3 L4"): m_trk(8,"V15 @1  P3 L4")
220 /*
230 owari=0 : s=0
240 for i=4 to maxtr : save_n(i)=saisho_go() : next
250 /*
260 while nokori(maxtr)
270   if s>10 then {
280      if (chosi=0)and(chord="C")  then break
290      if (chosi=1)and(chord="Am") then break
300   }
310   C_set(1,chord)
320   for i=4 to maxtr
330      old_n=save_n(i) : oku=save_oku(i) : old_oku=oku
340      print shosetu(i,chord);"  ";
350      save_n(i)=old_n : save_oku(i)=oku
360   next
370   if chosi=0 then chord=nxt_cod_maj(chord) else chord=nxt_cod_min(chord)
380   for i=4 to maxtr : save_n(i)=takasa(save_n(i),chord) : next
390   print
400   s=s+1
410 endwhile
420 /*
430 color 2 : print chord, : color 3 : end_C(1,chord)
440 for i=4 to maxtr
450   oku=save_oku(i) : old_oku=oku : tmp_str=nnn(1,saisho_go())
460   m_trk(i,tmp_str) : print tmp_str;"  ";
470 next
480 print
490 /*
500 m_play(1,2,3,4,5,6,7,8)
510 /*
520 end
530 /*
540 /* コードを設定する
550 /*
560 func C_set(t,c;str)
570 color 2: print c, : color 3
580 if c="C" then {
590   m_trk(t,"cccc") : m_trk(t+1,"eeee") : m_trk(t+2,"gggg")
600 } else if c="Dm" then {
610   m_trk(t,"dddd") : m_trk(t+1,"ffff") : m_trk(t+2,"aaaa")
620 } else if c="Em" then {
630   m_trk(t,"eeee") : m_trk(t+1,"gggg") : m_trk(t+2,"bbbb")
640 } else if c="F" then {
650   m_trk(t,"ffff") : m_trk(t+1,"aaaa") : m_trk(t+2,"<cccc>")
660 } else if c="G" then {
670   m_trk(t,"gggg") : m_trk(t+1,"bbbb") : m_trk(t+2,"<dddd>")
680 } else if c="Am" then {
690   m_trk(t,">aaaa<") : m_trk(t+1,"cccc") : m_trk(t+2,"eeee")
700 } else if c="Bdim" then {
710   m_trk(t,">bbbb<") : m_trk(t+1,"dddd") : m_trk(t+2,"ffff")
720 } else { /* E
730 /* E
740   m_trk(t,"eeee") : m_trk(t+1,"g+g+g+g+") : m_trk(t+2,"bbbb")
750 }
760 endfunc
```

します。先の例では，１小節には８－７＝１の長さだけ足りないことになりますから，８分休符をひとつ付ければよいことがわかります。しかし，メロディのなかに休符ばかりが現れるのは困りますから，こういった場合でもあと数回(ここでは３回)長さを求めてみて，それでもちょうど８にならない場合は休符を埋めることにします。先の例では，あと３回のうちに８分音符が出れば８分休符は入れないことにします。このようにして，

・コードを決定する。
・コードに対応する１小節分のメロディを決定する。

という操作を繰り返すことで曲を作っていくのです。曲の中間部は以上のようにして作ることができますが，問題は始まりと終わりです。

この作曲プログラムではリスト６と同様に，長調ならばＣのコードで，短調ならばAm のコードで始まることにします。また，始まりの音は長調ならばド(C)，ミ(E)，ソ(G)のどれか，短調ならばド(C)，ミ(E)，ソ(G＃)，ラ(A)のどれかで始まることにします。出現する確率は，リスト７と同様に，長調のときは，

C＜E＜G

短調のときは，

C＜E＜G＃またはAm

の順にしますが，それほど意味はありません（C，Amのコードを構成する音です)。

曲の終わりは，本来の作曲では難しいところですが，ここでは曲の最初の音と同じ選び方をした音を全音符で鳴らすだけにします。ただ，いきなり終わったのでは曲がぶっち切れた感じがしますから，ある程度曲を作った段階（たとえば10小節以上を作った段階）でC（長調の場合）または Am (短調の場合)のコードになったときに，全音符を鳴らして終わることにします。このとき，最後の音はひとつ前の音とまったく無関係になります（オクターブは同じにします）が，硬いことはいわないようにしましょう。

2）プログラム

リスト８が乱数による自動作曲プログラムです。リスト８ではチャンネル１～３でコードを演奏し，チャンネル４～８でメロディを演奏するようにしています。チャンネル４～８のメロディは同じコード進行上に作られてはいますが，お互いに無関係なメロディを演奏しています。しかし，同時に聞くと結構調和しているように聞こえるんですよ，これが（にぎやかなのが好きな

```
 770 /*
 780 /* おわりのコード
 790 /*
 800 func end_C(t,c;str)
 810   if c="C"  then m_trk(t,"c1") : m_trk(t+1,"e1") : m_trk(t+2,"g1") : retur
n()
 820   m_trk(t,">a1") : m_trk(t+1,"c1") : m_trk(t+2,"e1")
 830 endfunc
 840 /*
 850 /* つぎのコード（長調）
 860 /*
 870 func str nxt_cod_maj(c;str)
 880   if c="C" then {
 890           if ransu(2)=1 then return( either("F","G") )
 900           return( either("Am","Em") )
 910   } else if c="Dm" then {
 920           if ransu(2)=1 then return( "G" )
 930           return( "Am" )
 940   } else if c="Em" then {
 950           if ransu(2)=1 then return( "F" )
 960           return( either("Am","G") )
 970   } else if c="F"  then {
 980           if ransu(2)=1 then return( either("C","G") )
 990           return( either("Dm","Em") )
1000   } else if c="G"  then {
1010           if ransu(2)=1 then return( either("C","Am") )
1020           return( "Em" )
1030   } else { /* Am
1040           if ransu(2)=1 then return( either("F","Dm") )
1050           return( either("G","Em") )
1060   }
1070 endfunc
1080 /*
1090 /* つぎのコード（短調）
1100 /*
1110 func str nxt_cod_min(c;str)
1120   if c="Am" then {
1130           if ransu(2)=1 then return( either("Dm","E") )
1140           return( either("F","Bdim") )
1150   } else if c="Bdim" then {
1160           return( "E" )
1170   } else if c="Dm"   then {
1180           if ransu(2)=1 then return( either("Bdim","Am") )
1190           return( "E" )
1200   } else if c="E"    then {
1210           return( either("Am","F") )
1220   } else { /* F
1230           if ransu(2)=1 then return( either("Dm","E") )
1240           return( either("Bdim","Am") )
1250   }
1260 endfunc
1270 /*
1280 /* どちらかの文字列を選択
1290 /*
1300 func str either(a;str,b;str)
1310   if int(rnd()*2) then return(a) else return(b)
1320 endfunc
1330 /*
1340 /*  コードに対応したメロディーをつくる
1350 /*
1360 func str shosetu(trk,chord;str)
1370   int l,n,sum_nag,tmp_nag,i
1380   str mm,res
1390   l=nagasa() :sum_nag=8 ¥ l
1400   res=nnn(l,old_n) : m_trk(trk,res)
1410   while sum_nag<8
1420        l=nagasa() : tmp_nag=8 ¥ l
1430        if (tmp_nag+sum_nag)>8 then {
1440           for i=0 to 2
1450                l=nagasa() : tmp_nag=8 ¥ l
1460                 if (tmp_nag+sum_nag)<9 then break
1470            next
1480            if (tmp_nag+sum_nag) >8 then break
1490        }
1500        sum_nag=sum_nag+tmp_nag
1510        n=takasa(old_n,chord) : old_n=n
1520        mm=nnn(l,n) : res=res+mm : m_trk(trk,mm)
1530   endwhile
1540   if sum_nag<8 then {
1550        for i=1 to (8-sum_nag) : m_trk(trk,"R8")
1560        res=res+"R8" : next
1570   }
1580 return(res)
1590 endfunc
1600 /*
1610 /*  最初と最後の音
1620 /*
1630 func int saisho_go()
1640     switch ransu(3)
1650     case 0: return(0)
1660     case 1: return(2)
1670     default:if chosi=0 then return(4)
1680             if int(rnd()*2) then return(5) else return(4)
1690     endswitch
1700 endfunc
1710 /*
1720 /* 音の高さと長さをMMLに変換
1730 /*
1740 func str nnn(l,n)
1750    str r,s=""
```

んです)。

リスト8のプログラムは，基本的には，リスト7の自動作曲プログラムと同様に曲を作ります。しかし，5つのチャンネルの演奏データをfor〜nextループで順々に作っていくため，チャンネルごとのひとつ前のデータをsave_n(音名)，save_oku(オクターブ)といった配列に記憶しています。これがプログラムを読みにくくしている原因ではありますが，気を入れて読んでもらえばそれほど難しいプログラムでないことがわかると思います。ということで，プログラムの解説はこれ以上しなくてもよいでしょう。

なお，メロディの演奏をひとつのチャンネルだけで行いたい場合は，プログラム中のmaxtrという変数の初期値を4（最大4チャンネルを使うという指定）にしてください。同時に演奏しているときにはなかなか聞き取ることのできなかった，X68000の作るメロディをじっくりと聞くことができますよ。リスト8のプログラムによって作られる曲のうち，1チャンネルについての冒頭部分を図10に示しておきます。どうです。なかなかそれらしい出来になっているでしょう。

おわりに

MMLというと，既存の曲を入力してただ聞くだけという楽しみ方しかないようにも思われがちです。しかし，乱数を使って「それらしい」曲を作り出していくことも結構面白い楽しみ方ではないでしょうか。今回は単純なコード進行と，連続する2音の関係のみを考慮して曲を作ってみましたが，もっと複雑なコード進行やメロディの生成法を使うとか，特殊な音階を使うとか，いろいろな工夫をすることで，本当に感動的な曲を作るのも夢ではないと思います。皆さんもいろいろ試してみませんか。

さて，来月はAD PCMに挑戦してみたいと思います。なにが出るかお楽しみ（実は，まだなーんにも考えていない)。それでは，来月までさようなら。

〈参考文献〉

1) 河西保郎，『やさしい作曲のABC・ソングライターへの近道』，ケイ・エム・ピーkmp，1987年。
2) 池田寛(他)，『最新音楽用語事典』，リットーミュージック，1987年。
3) 祝一平，『試験に出るX1・ハードウェアのフルコース』，第11章，日本ソフトバンク，1987年。
4) 山本善介(編)，『ピアノ曲集・機動戦士ガンダム』，東京音楽書院，1988年。


```
1760    if (chosi=1)and(onkai(n)=7) then s="+"
1770    r=nn(onkai(n))+s+str$(l)
1780    if oku<>old_oku then r="O"+str$(oku)+r
1790    old_oku=oku
1800    return(r)
1810 endfunc
1820 /*
1830 /* 長さ 4 > 8 > 2 > 1 の順
1840 /*
1850 func int nagasa()
1860   switch ransu(4)
1870   case 1: return(2)
1880   case 2: return(8)
1890   default:return(4)
1900   endswitch
1910 endfunc
1920 /*
1930 /* 高さ 差が 3 > 2 > 4 > 0 > 6 > 5  の順
1940 /*
1950 func int takasa(mae,c;str)
1960   int i
1970   switch ransu(6)
1980   case 0: i=4 : break
1990   case 1: i=5 : break
2000   case 2: i=0 : break
2010   case 3: i=3 : break
2020   case 4: i=1 :break
2030   default:i=2 : break
2040   endswitch
2050   if oku>4 then {
2060      if ransu2(2) then i= -i
2070   } else if oku<4 then {
2080      if ransu2(2)=0 then i= -i
2090   } else {
2100      if int(rnd()*2) then i= -i
2110   }
2120   i=i+mae
2130   if i<0 then {
2140      oku=oku-1 : i=i+7
2150   }else if i>6  then{
2160      oku=oku+1 : i=i-7
2170   }
2180   if oku=0 then oku=1
2190   if oku=8 then oku=7
2200   return( in_cod(i,c) )
2210 endfunc
2220 /*
2230 /* コードの中の音
2240 /*
2250 func int in_cod(n,c;str)
2260   int i,j,k
2270   dim char cc(2)
2280   if c="C" then              { cc(0)=0 : cc(1)=2 : cc(2)=4
2290   }else if c="Dm"   then { cc(0)=1 : cc(1)=3 : cc(2)=5
2300   }else if c="Em"   then { cc(0)=2 : cc(1)=4 : cc(2)=6
2310   }else if c="F"    then { cc(0)=3 : cc(1)=5 : cc(2)=0
2320   }else if c="G"    then { cc(0)=4 : cc(1)=6 : cc(2)=1
2330   }else if c="Am"   then { cc(0)=5 : cc(1)=0 : cc(2)=2
2340   }else if c="Bdim" then { cc(0)=6 : cc(1)=1 : cc(2)=3
2350   }else                  { cc(0)=0 : cc(1)=2 : cc(2)=4 }
2360   for i=0 to 2
2370      if cc(i)=n then break
2380   next
2390   if i<3 then return(n)
2400   if ransu(2)=0 then return(n)
2410   i=abs(n-cc(0)) : j=abs(n-cc(1)) : k=abs(n-cc(2))
2420   if i>j then i=j : j=cc(1) else j=cc(0)
2430   if i>k then return(cc(2)) else return(j)
2440 endfunc
2450 /*
2460 /*  乱数の初期化
2470 /*
2480 func rnd_init()
2490   str t : int h,m,s
2500   t=time$
2510   h=val(left$(t,2)) : m=val(mid$(t,4,2)) : s=val(mid$(t,7,2))
2520   randomize((h*m*s)and &H7FFF)
2530 endfunc
2540 /*
2550 /*  残りトラック
2560 /*
2570 func int nokori(n)
2580   int p=-1,i
2590   for i=1 to n : p=p and (m_free(i)>50) :next
2600   return(p)
2610 endfunc
2620 /*
2630 /*  乱数 (0 に近い値ほど出にくい)
2640 /*
2650 func int ransu(n)
2660   return( int(sqr(rnd())*n) )
2670 endfunc
2680 /*
2690 /*  乱数 (その2)
2700 /*
2710 func int ransu2(n)
2720   return( int(sqr(sqr(rnd()))*n) )
2730 endfunc
```

AI開発セット/OS-9/X68000 Sampling PRO-68K

X68000あなたの知らない世界

X68000

ビジネスショウで，ついに姿を現したOS-9/X68000。今回はX68000用マルチタスクOSとウィンドウシステム，そして待ちに待ったAD PCM用ツール，Sampling PRO-68Kについての速報として製品概要を紹介していきます。

AI開発セット

今年のビジネスショウで発表されたX68000関係の新製品を紹介してみましょう。まずは，LISPベースのエキスパートシステム開発ツールです。このプログラムはSTAFF LISPというスタンダードLISP系列の高機能LISPで書かれています。最近は「～系のLISP」というのははやりませんが，ほとんど専用LISPマシンと同等かそれ以上の性能を示すという高性能ぶりが魅力です。日本ではPC-9801の68000ボード用にCP/M-68Kバージョンが発売されていますが，198,000円とちょっと個人では手の出ない価格でした。X68000版では安価に発売されることを祈りましょう。

エキスパートシステム構築ツールによるサンプルとしてビジネスショウのデモではマイクロマウスが画面上の迷路を解いてみせたり，画面に表示されたハノイの塔を解いたりといった簡単なデモを実演していました。

数式処理システムREDUCEはメモリさえ積んでやれば大型機以上の性能を発揮することでしょう（広告によるとMC68000と2MバイトRAMで大型機並みだそうな)。

こういった記号処理言語などではシステムの主記憶容量がそのまま性能に反映されてきます。その点ではMC68000プロセッサは向いているといえます。特にX68000は主記憶最大12Mバイトまで拡張できますので，その気になれば超本格的な処理が可能です。ひと頃は「PC-9801でREDUCEがとりあえず走る＝凄い」といわれていたのですから，時代も変わったものですね。

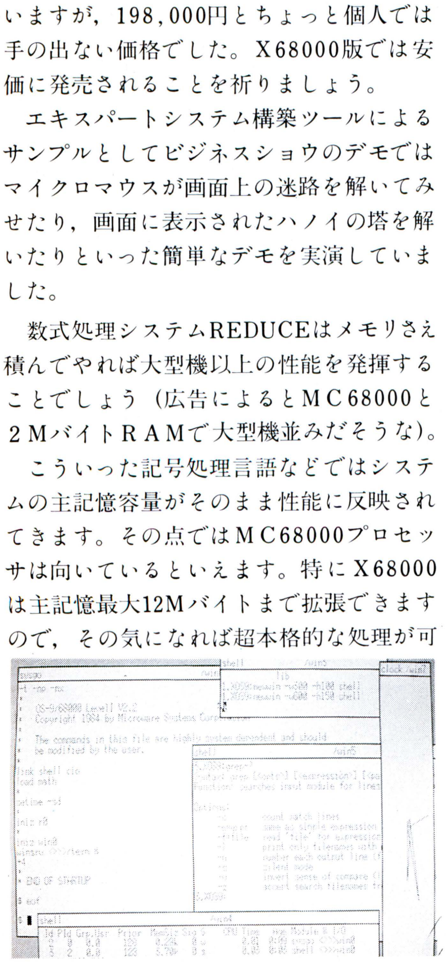


写真1　これがOS-9だ

OS-9/X68000登場

昨年夏あたりから「秋葉原の某所ではX68000でOS-9が走っている」という噂がひそかにささやかれていましたが，それがついにシャープブランドでOS-9/X68000として発売されることになりました（開発はマイクロウェア・ジャパン)。ビジネスショウのシャープブースでは参考出品ながら，4台のマシンでOS-9がデモを行っていました。うち2台はアークネットという独自のインタフェイスでネットワークサーバに接続されており，X68000にもようやくワークステーションといった風情が出てきたようです。

画面にはPERSONAL WINDOWと呼ばれるマルチウィンドウが展開され，その上で日本語UNIFY（もとはUNIX用のデータベース）やDYNACALCといったビジネスソフトが並行動作していました。なかなかの迫力です。

OS-9とHuman

OS-9というものにあまり知識のない方が多いと思いますので簡単に解説してみましょう。OS-9はマイクロウェア社が開発した68系のOSです。もとはMC6809用のOSだったのですが，現在では16ビット版のOS-9/68000，32ビット版のOS-9/68020などが作られています。

Humanとは違うOSだからといって操作法や概念などがまったく異なるわけでもありません。UNIX以降のOSはすべてUNIXの影響を受けているといわれるとおり，OS-9もHumanの元となったMS-DOSもUNIXを意識して開発されたものなのです。機能的に見ても階層化ディレクトリ，ユニファイドI/O，リダイレクト，パイプライン処理など，UNIXから受け継いだものについてはMS-DOS（Human68k）とほぼ同じです。

まず，Humanとのファイル互換性ですが，ディスクフォーマットが違いますのでそのままのかたちでは互換性はありません。しかし，OS-9にはMS-DOS，PC-DOS（IBM PCのフォーマット）のディスクを読み書きするためのデバイスドライバが標準装備されるため，OS-9上では特に問題はないでしょう。ただし，Humanのビジュアルシェルなどで OS-9用のディスクを使うとシステムがフォーマットされていないディスクと判断して，フォーマットしようとしますので注意が必要でしょう。これにはHuman用のデバイスドライバを変更するなど，メーカー側でなんらかの対処が望まれるところです。

マルチタスクの威力

いちばんの違いはマルチユーザー，マルチタスクに対応しているかどうかということです。たとえば，マルチユーザーに対応しているため，OS-9ではコマンド用のパスとデータ用のパスが設定されており，各ユーザーは自分に割り当てられたパスの中だけで作業するようになっています。

システムの負担を最低限にしてマルチタスクを実現するため，OS-9上のプログラムのほとんどがリロケータブル，リエントラントに作られています。リロケータブルというのは「メモリ上のどこに置いても動作する」ということですが，OS-9では実行時に新しいプログラムを組み込む（ダイナミックリンク）際のアドレス変換のオーバーヘッドをなくすため，最初からリロケータブルなプログラムのみを扱うようになっているわけです。MC68000ではリロケータブルなプログラムを書くことはそう難しいことではありませんが，実行速度などではやや不利となります。

リエントラントとはいわば「複数のプログラムから使用できる」という意味です。たとえば，BASICをマルチタスクで実行

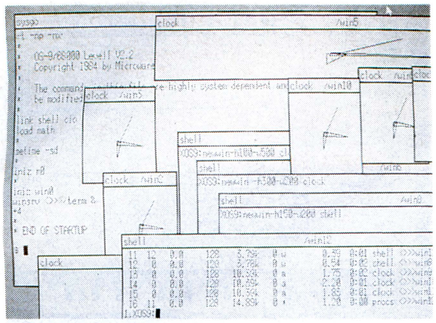


写真2 マルチタスクの例

する場合，ふつうはBASICが実行されるたびにディスクからプログラムがロードされるわけですが，OS-9ではメモリ上にすでにそのプログラムがある場合は読み込みをキャンセルしてメモリ上のプログラムをそのまま使ってやろうというのです。そのためには，プログラムが使用するレジスタやワークエリアはプログラム本体とは完全に分離して，呼び出したプログラム側でスタックなどを使って管理しなければなりません。アセンブラで開発する場合は少々注意が必要ですが，CやBASIC，Pascalなどの高級言語を使用するときはコンパイラが自動的にそのようなコードを生成してくれるので，ユーザーは特にリエントラントを気にする必要はありません。

そして，こういった「作法」に従ったプログラム群がOS-9を構成していくのです。もともとOS-9のカーネルは非常に小さなものです。これに，デバイスドライバ，シェル(command. xにあたるもの)，コマンド群などのモジュールがリンクされて一人前のシステムのできあがり。OS-9の世界ではモジュール化ということは本質的な意味を持ちます。MS OS/2が涙ぐましい努力を重ねてマルチタスクを実現しているのに対し，OS-9では最初からマルチタスクOSとして設計されているため，非常にスッキリしたシステム構成を保っています。

マルチタスクによる恩恵はいろいろと考えられるでしょう。プリントアウト中にコンパイラやエディタが動けばそれなりに快適でしょう。ワープロやデータベースでタスク間のデータ通信ができればあっという間に統合化ソフトですし，ゲームなどでも威力を発揮しそうですね。

また，ローカルエリアネットワークにも対応したシステムが用意される予定です。これはビジネスショウにも出展されていたアークネットによるものですが，転送レートは2.5 Mbpsという速度ですので，ネットワークを構築した場合の違和感もほとんどないでしょう。UNIXのネットワークの場合，ひとつの大型機を中心に各UNIX端末からリモートログインして大型機をいじめる（？）という使い方が一般的ですが，このOS-9ネットの場合はこれとは逆に各マシンをI/Oとして扱います。リモートログインのようなことはできませんが，各端末に接続された周辺機器を自由に扱え，もちろん端末間のデータ通信もできます。

PERSONAL WINDOW

X68000ユーザーには待望のウィンドウ環境が提示されました。X68000にはこれまでに，WINDEX，Kamikaze，VS.Xなど独自のウィンドウシステムを使ったソフトウェアは登場していましたが，そういったウィンドウ環境がユーザーに対して開かれたことはありませんでした。メーカーからもこういった環境についてなんの指導もなく，ある者はウィンドウドライバ制作を夢見たり，また，ある者はMacintoshをうらやむといった状況だったのです。

PERSONAL WINDOWはUNIX上で広く使われているX-WINDOWのコンセプトをベースにした専用のウィンドウシステムです。これでは，1024×1024の仮想画面をフルに使い，マウスカーソルで自由に表示エリアを指定することができます。このウィンドウは基本的にテキストベースのものですので，グラフィックには対応していません。しかし，X68000のテキストVRAMはふつうの16ビット機のグラフィックに相当する能力を持ったビットマップ方式ですので，これだけでも相当の表現力を発揮できるでしょう。

ウィンドウ操作にはパソコンでは初めて，マウスカーソルのある領域がアクティブウィンドウになるという「アクションリージョン」方式が採用されています。WINDEXやVS.XなどのMacintosh式ウィンドウではいちばん上に表示されているウィンドウがアクティブウィンドウ（ユーザーがアクセスするウィンドウ）として扱われ，下敷きになっているウィンドウには飾り程度の役割しかなかったわけですが，マルチタスクで動くPERSONAL WINDOWではアクティブウィンドウ以外のウィンドウでもプログラムの実行動作を表示し続けます。また，ほかのウィンドウが半分かぶさっているような状態でも，マウスカーソルがその領域にあれば，そのままキー入力が可能です。

ちなみに，写真で画面上に開かれているウィンドウの大きさは80×25字（640×400ドット），ふつうの端末1画面分のものが余裕をもって並びます。画面写真のものはプロトタイプですのでタイトルバーまわりにもスイッチ類は備えられていませんが，製品版ではもっと充実するとのこと。これらの環境はすべてユーザーに開放されます。

また，変わった機能として，X68000の4枚のテキストプレーンを4つの表示機器とみなして切り換えできるデバイスドライバなどもサポートされるようです。こうなると使い方に困ってしまいそうですね。

言語・アプリケーション

OS-9のひとつの目玉といえば，BASIC09でしょう。BASIC09は構造化を強く意識したBASICで，コンパイラ仕様となっています。OS-9はもともとBASIC09を動かすために作られたOSだといわれますが，肝心のBASIC09は残念ながら付属してきません。従来のOS-9/68000用のBASIC09（そのほかの言語も）はそのまま実行可能ですが，さらにX68000の機能をフルサポートした専用版が現在開発されています。

OS-9用のCはUNIXのCに準拠したものに専用ライブラリを加えたものでUNIX用のソースなら90%以上そのまま動作するらしいのですが，これも別売の予定となっています。OS-9には標準でMAKEが装備されているほか，Cにはソースレベルデバッガがついていますので開発環境としては，Human以上といえるかもしれません。MS-DOSのCからはXCに，UNIXのCからはOS-9のCにと使い分ければX68000のソフトウェア資産も急増しそうですね。

また，懸念された日本語処理関係はほぼHumanとコンパチのものとなる予定。標準装備されるフロントプロセッサは，ASK.68KをOS-9用にリロケータブル/リエントラントにしたもので，若干高速化されているとのこと。X68000独自のソフトウェアとしてはAV RIDERという，FM音源，ADPCM，グラフィック機能を統合的に扱うシェルのようなものが予定されています。

OS-9自体の発売時期はまだはっきり決まっていませんが，秋頃には登場するはずです（価格については未確認情報ですが3万円を切るのではないかとの噂も。ちなみにOS/2はR70用で58,000円）。

最後に

さて，写真3を見てください。これがビジネスショウでも展示され，意味不明といわれたOS-9のデモです。宇宙を飛ぶ戦闘機（？）のスクリーンに映った映像が次第に拡大されていくというもので，ちょっと見た目にはHumanでも簡単にできそうなデモなのですが，これにはひとつの主張が込

められています。というのも，これらはすべてOS-9のファンクションコールのみを使ったデモなのです。

X68000用に現在発表されているゲームなどのアプリケーションのほとんどはHumanのファンクションコールやIOCSを無視して，直接VRAMなどのI/Oを操作するなどの高速化がなされています。無論，リエントラントに書かれたものなどは存在しないでしょう。これでは将来X68000の32ビット版が発売されても「スペースハリアーがウィンドウで動く」といったことは到底望めそうもありません。

一般にOS-9の世界では，このようにアプリケーションがデバイスドライバやファンクションコールをとおさずにI/Oに触れることは許されていません。あの宇宙船のデモはOS-9のファンクションコールだけでも，これだけのことができるというデモだったはずなのですが……。

ふつうのユーザーがコンピュータを使用する際には，マルチタスクよりもシングルタスクで使うことのほうが多いのは事実でしょう。実際，これまでのパソコンはシングルタスクでも用が足りていたことを思えば，これまでのHumanでもたいていのことでは困ることはありません。しかし，マルチタスクが導入されることでまた新しいコンピューティングの世界が開けてくることでしょう。

ただ，気をつけてほしいのはOS-9とHumanを比べた場合，単純にマルチタスクだからOS-9が優れているというわけではないということです。MS-DOSV2.2に比べればHumanは動作の安定性など，非常に優れた面もあり，そもそもシングルユーザー/シングルタスクを目指したものと，マルチユーザー/マルチタスクを目指したものとでは設計思想が根本的に異なるのです。

とにかく，これでX68000にHuman68k，CP/M-68K，OS-9/68000と3種類のOSが揃いました。どれを選ぶかはユーザー次第。なかなか楽しくなってきましたね（さぁて，次はUNIXだ）。

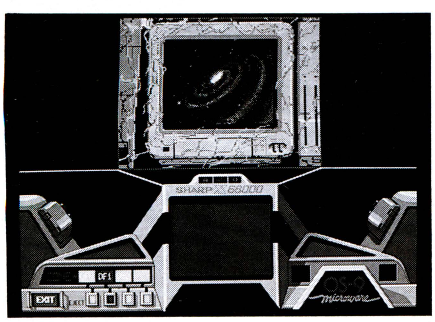


写真3　問題のデモ

Sampling PRO-68K

SOUND PRO-68K，MUSIC PRO-68Kに続く，シャープブランドの音楽ソフトがこのSampling PRO-68Kです。前2作がFM音源用のユーティリティだったのに対して，このソフトはX68000のAD PCMを最大限に活用しようというコンセプトで開発されています。

楽器でサンプリングというとカシオのサンプルトーンやフェアライトCMIが有名ですが，X68000に使用されているPCM用のチップはこういった音楽用に開発されたものではなく，どちらかといえば留守番電話などに向いた石です。まず，サンプリングは4ビットの固定長デルタPCMですのでそれほどの音質は望めません。加えて，ハードウェア自体には周波数変調の機能がないので，音楽として使う場合にはソフトウェアの負担が非常に大きなものとなります。

しかし，それでもなおサンプリング音源は従来のFM音源やSSGでは不可能だった表現を可能にします。「誰も聞いたことのない音を作りたい」というのがシンセサイザの原点であるならば，「音＝空気の振動」をもっとも明確に提示してくれるPCM方式が秘めた可能性を見逃すわけにはいかないでしょう。

このSampling PRO-68Kでは，サンプルした音を全/半角1文字のラベル名で管理します。一般的には“あ”から“ん”までの50音などをそのまま1音ずつ登録しておいてしゃべらせる，といった用途が多いでしょう。ひとつのファイルは最大で約700Kバイト，300ラベル（約200秒）まで登録できます。もちろん，ひとつのラベルに長いデータを入れてもかまいません。

どうしても楽器として使いたいという方は使用するキーすべてにラベルをつけるという荒技もできます。シンセでもマルチサンプリングは常識なのだから，手持ちの楽器から1音ずつ音を録るのもよいでしょう。これなら，実用になる音が出ます。

基本的な使い方

では順を追ってSampling PRO-68Kの使い方を見てみましょう。まず，データのサンプリングを行います。これにはBASICなどと同様そのままの音を取るモードと音量があるレベルに達したら自動的に取り込みを開始するトリガーモードがあります。無論，トリガーレベルは簡単に設定可能ですし，最近のサンプリングキーボードで見

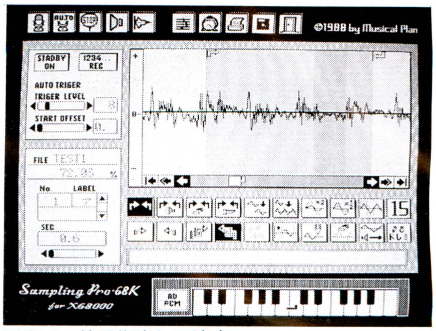


写真4　範囲指定して演奏

られるスタートオフセット機能も設定できます。これはトリガーモードで立ち上がり途中の音が頭切れになってしまうのを防ぐため，トリガーレベルに達する前の状態を一定間隔で保存する機能です。

ラベル名を指定してサンプリングを行ったら，その音を編集してみましょう。まず，プルダウンメニューで全体の波形を表示させ，エディットしたいあたりをクリックしてみます。するとエディット用のウィンドウにその部分が詳しく再表示されます。ここで波形のインサート/デリート，振幅調整，波形合成，手書き修正などの操作が可能です。修正されたデータは画面上のミュージックキーボードまたは実際のキーボード（コンピュータの）で音程を変えて演奏し，耳で確認することもできます（音程によってはノイズっぽくなる）。この場合，一定範囲だけを演奏する，一定範囲を繰り返し演奏するといった操作により，適当な区間を選んでやれば，きわめて短いデータでその音色を表現できるわけです（うまくいけば）。

そのほかの機能

このようにして作られたサウンドデータはBASICやMUSIC PRO-68Kで作られた音楽データとリンクしたり，目覚まし時計のようにタイマーセットで起動することができます。また，ユーティリティとしてOSやBASICのエラー番号に応じてエラー内容を音声出力するためのツールや，キー入力やリスト出力を読み上げてくれるツールが付属します。このツールで作ったデータはOSレベルで使用できるわけです。

ディスクは2枚組で1枚はシステム，もう1枚はデータディスクとなっています。自分の声を再生してもちょっと不気味だという人は付属の音声データを使用しましょう。若くて美しい（そう信じよう）女性の声が登録されています（モーニングコールのデータも入っているぞ）。発売予定は7月初め。君のX68000が歌い出す日も近い！

（中野　修一）

Object oriented

実用(?)オブジェクト指向のゲームプログラミング——第7回

完結のスネークオブジェクト

Hamaguchi Isamu
浜口　勇

大胆にもZ80でオブジェクト指向によるリアルタイムゲームのプログラミングをという試みで始まった連載だが，苦闘の末ようやくプログラムを走らせるところまでこぎつけることができたようだ。今回と次回で完成したプログラムを発表することにしたい。

さて，今月でプログラムのほうは一応動くようになった。これで，まったく使いものにならないということもないということになる(やれやれ)。

最終的には全体的に変更しなければならない部分が出てきたので全部のリストをリスト1に掲載しよう。プログラム全体の構成は今まで説明したのと変わっていないのでそちらを参照してもらうことにして，今回はあまり細かいことをグチグチ書いていっても理解できなくなる可能性があるのでできるだけわかりやすい話題を中心に取り上げていってみよう。

さて，ここで連載初回からの疑問をもう一度ぶり返すことになる。

「オブジェクト指向というのは何なのか。それは容易に実現できるのか。本当に役に立つのか」

この3つの疑問に対する解答というのは果たして発見できたのか？

最初のテーマ

まずオブジェクト指向というのは何なのか？　という疑問であるが，答えは「データとプログラムをまとめて扱うことによってシステムを抽象化する試み」などという言葉でお茶を濁したほうが賢明なようだ。なぜなら，オブジェクト指向にとって絶対神ともいえたSmalltalkがその権威を失いつつあるためにその定義も混沌の中に落ち込みつつあるからだ。

たとえばクラスの問題というのがあって，「クラスというのは特別なインスタンス(つまりオブジェクト）なのでインスタンスさえあればクラスは不要である」という意見があるようである。普通のインスタンスとクラスといった場合分けがあるのはシステムを複雑化するだけだからやめたほうがいいというわけだ。

では今回のシステムになぜクラスがあるのかというと，実はROM化といった問題が残されているからである。たとえばあるプログラムをROM化しようとした場合，そのプログラムの中には当然ROMの上に乗っている部分とRAMの上にある部分が出てくる(図1)。ROMは書き換えができないからROMの上にあるデータというのは変数ではなく定数ということになるわけだ。

さて，もしもインスタンスだけしかないシステムの場合ROMの上にある変数とRAMの上にある変数という2つの状態が混在してわかりにくい。しかしクラスがあるならばクラスの変数というのはすべてROMの上にあって書き換えができないということにしてしまえばいいので簡単である。このようにある状況ではクラスというのは大変有用なのだ。

ではなぜクラスは必要ないという論議が出るかというと研究者はプログラムをROM化したりしないということなのだろう。つまりそれぞれの分野によって必要とされているオブジェクト指向が違うのである。

オブジェクト指向の可能性

次に2番目の答えは簡単である。実現できる。この記事の後ろに実例が載っている。ただしプログラム中によい例とはいえない部分が多く見られるのも事実だ。

sheadやbackgなどのプログラム中には定数の使用が結構見られるのだけど，これはできればクラス変数にすべきだろう。なぜクラス変数になっていないかというと，Z80というCPUの特質により，クラス変数にアクセスするのがとても面倒なためだ。sheadやbackgは他のクラスのスーパークラスにはなっていないために今回は特にクラス変数を使う必要はなく，デバッグの容易さからクラス変数はあまり使わなかった。

本当に役に立つのかという疑問に対しては，「少なくともビデオゲームプログラミングというジャンルにおいてはかなりの意味を持つ可能性がある」と答えたい。高級言語や洗練された命令体系を持ったCPUとは結構相性がよいようである。

ではZ80とは？　と聞かれるとかなり苦しいものがある。変数を操作するのに手間がかかり過ぎ，プログラムが直観的でなくなるためバグが発生しやすくなるのである。今回のプログラムも2Kバイトないのにバグのオンパレードであった。じゃZ80は使いものにならないのか，というとそんなことはない。高級言語を使えばよいのだ。最近の高級言語は素晴しいコードを出してくれるのでかなり開発は容易になる。

僕個人としては,「使いものにならない！」と力説されても細かいところでチョコチョコと使っているので,「でも一応使っているんですけど」と反論するしかない。この連載もCなんかを対象にするとかなりわかりやすくなったかもしれないが，なにしろCユーザーなど数えるほどしかいないということでその線はあきらめざるをえなかったというところだ。

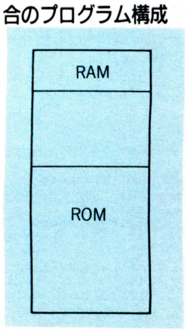
さて，最後にプログラムも揃ったということでリストのアセンブルのしかたについて説明しておこう。まずclassmがあるならば，それを使う，なければ.defリストを一生懸命入力する。今回の記事だけならば.defファイルを入力したほうが早いだろう。

次に.macファイル等を入力する。この場合すべてのファイルとM80はM80のバグのために(インクルードに関するバグ)同一ドライブ上に揃っていなければならない。

そして最後にruntime.relが先頭にくるようにリンクすればよい。面倒臭いのでまとめてリスト2に載せておく。

次号は少し落ち着くので最後として，ダンプリストの掲載と今までのおさらい，Cへの拡張などやるつもりなのでよろしく。

図1　ROM化した場合のプログラム構成



リスト1-1 class.def

```
================= CLASS.DEF =================
selfclass       macro   arg
        ld      hl,arg
        call    _call##
        endm
selfinstance    macro   arg
        ld      hl,arg
        call    _call##
        endm
callclass       macro   arg
        ld      hl,arg
        call    _call##
        endm
callinstance    macro   arg
        ld      hl,arg
        call    _call##
        endm
```

リスト1-2 runtime.mac

```
================= runtime.mac ==============
        .Z80
        dseg
wtop::  ds      2
work:   ds      32*1024
        cseg
main@::
        di
        ld      d,00
        ld      e,18
        ld      hl,screentable
scloop:
        ld      bc,1800h
        out     (c),d
        inc     bc
        ld      a,(hl)
        out     (c),a
        inc     hl
        inc     d
        dec     e
        jp      nz,scloop
        ld      bc,1a03h
        ld      a,0dh
        out     (c),a
        ld      de,work
        ld      (wtop),de
        ld      bc,1024
wloop:
        ld      hl,32
        add     hl,de
        ex      de,hl
        ld      (hl),e
        inc     hl
        ld      (hl),d
        dec     bc
        ld      a,b
        or      c
        jp      nz,wloop
        jp      start##
_call::
        push    de
        push    hl
        ld      h,b
        ld      l,c
        ld      e,(hl)
        inc     hl
        ld      d,(hl)
        inc     de
        inc     de
        ex      de,hl
        ld      e,(hl)
        inc     hl
        ld      d,(hl)
        pop     hl
        add     hl,de
        ld      e,(hl)
        inc     hl
        ld      d,(hl)
        ex      de,hl
        pop     de
        jp      (hl)
        .radix  16
screentable:
        db      37,28,2d,34,1f,02,19,1c,
                00,07,00,00,00,00,00,00,
                00,00
        end
```

リスト1-3-a start.def

```
================= start.def =================
;       meta class
metaclass:
        dw      metaclass
        dw      classMethod
;       class var
cvarsize        equ     0
_backgnew       macro
        callclass       0
        endm
_backgalloc     macro
        callclass       2
        endm
_screennew      macro
        callclass       0
        endm
_screenalloc    macro
        callclass       2
        endm
_joypotnew      macro
        callclass       0
        endm
_joypotalloc    macro
        callclass       2
        endm
classMethod:
;       instance var
ivarsize        equ     0
@backgfreeobj   macro
        callinstance    0
        endm
@backgfree      macro
        callinstance    2
        endm
@backgnextobj   macro
        callinstance    4
        endm
@backglink      macro
        callinstance    6
        endm
@backgdemon     macro
        callinstance    8
        endm
@backgatcheck   macro
        callinstance    10
        endm
@screenfreeobj  macro
        callinstance    0
        endm
@screenfree     macro
        callinstance    2
        endm
@screenbox      macro
        callinstance    4
        endm
@screenputc     macro
        callinstance    6
        endm
@joypotfreeobj  macro
        callinstance    0
        endm
@joypotfree     macro
        callinstance    2
        endm
@joypotinp      macro
        callinstance    4
        endm
instanceMethod:
```

リスト1-3-b start.mac

```
================= start.mac =================
        include CLASS.DEF
        include START.DEF
start::
        ld      bc,screen##
        _screennew
        ld      bc,joypot##
        _joypotnew
        ld      bc,backg##
        _backgnew
        ld      bc,bwork##
        @backgdemon
        ret
        end
```

リスト1-4-a object.def

```
================= object.def =================
;       meta class
metaclass:
        dw      metaclass
        dw      classMethod
;       class var
mclass  defl    0
imethod defl    2
memsiz  defl    4
cvarsize        equ     6
_selfnew        macro
        selfclass       0
        endm
_selfalloc      macro
        selfclass       2
        endm
classMethod:
@@0001  equ     new
        public  @@0001
@@0000  equ     alloc
        public  @@0000
        dw      @@0001
        dw      @@0000
;       instance var
class   defl    0
ivarsize        equ     2
@selffreeobj    macro
        selfinstance    0
        endm
@selffree       macro
        selfinstance    2
        endm
instanceMethod:
@@0003  equ     freeobj
        public  @@0003
@@0002  equ     free
        public  @@0002
        dw      @@0003
        dw      @@0002
```

リスト1-4-b object.mac

```
================= object.mac =================
        include CLASS.DEF
        include OBJECT.DEF
object::dw      metaclass
        dw      instancemethod
        dw      ivarsize
alloc:  ld      hl,(wtop##)
        ld      e,(hl)
        inc     hl
        ld      d,(hl)
        dec     hl
        ld      (wtop##),de
        ex      de,hl
        ret
new:    _selfalloc
        ex      de,hl
        ld      (hl),c
        inc     hl
        ld      (hl),b
        dec     hl
        ex      de,hl
        ret
free:   ld      h,b
        ld      l,c
        ld      de,(wtop##)
        ld      (wtop##),hl
        ld      (hl),e
        inc     hl
        ld      (hl),d
        ret
freeobj:@selffree
        ret
        end
```

リスト1-5-a acter.def

```
================= acter.def =================
;       meta class
metaclass:
        dw      metaclass
        dw      classMethod
;       class var
mclass  defl    0
imethod defl    2
memsiz  defl    4
cvarsize        equ     6
_selfnew        macro
        selfclass       0
        endm
_selfalloc      macro
        selfclass       2
        endm
_supernew       macro
        call    @@0001##
        endm
_superalloc     macro
        call    @@0000##
        endm
_acternew       macro
        callclass       0
        endm
_acteralloc     macro
        callclass       2
        endm
classMethod:
@@0011  equ     new
        public  @@0011
        dw      @@0011
        dw      @@0000##
;       instance var
class   defl    0
linkup  defl    2
linkdw  defl    4
ivarsize        equ     6
@selffreeobj    macro
        selfinstance    0
        endm
@selffree       macro
        selfinstance    2
        endm
@selfnextobj    macro
        selfinstance    4
        endm
@selflink       macro
        selfinstance    6
        endm
@selfdemon      macro
        selfinstance    8
        endm
@superfreeobj   macro
        call    @@0003##
        endm
@superfree      macro
        call    @@0002##
        endm
@acterfreeobj   macro
        callinstance    0
        endm
@acterfree      macro
        callinstance    2
        endm
@acternextobj   macro
```

▶よかったあ。高橋君も浪人してたんだあ。6月号を見てホッとしました。だいたい予想はしていたが。間違っても予備校2年生とならないようお互いにがんばりましょう。

山田　純二　(18)　神奈川県

```
        callinstance    4
        endm
@acterlink      macro
        callinstance    6
        endm
@acterdemon     macro
        callinstance    8
        endm
instanceMethod:
@@0015  equ     nextobj
        public  @@0015
@@0014  equ     link
        public  @@0014
@@0013  equ     freeobj
        public  @@0013
@@0012  equ     demon
        public  @@0012
        dw      @@0013
        dw      @@0002##
        dw      @@0015
        dw      @@0014
        dw      @@0012
```

リスト1-5-b acter.mac

```
================== acter.mac ================
        include CLASS.DEF
        include ACTER.DEF
acter:: dw      metaclass
        dw      instancemethod
        dw      ivarsize

new:    _supernew
        ld      hl,linkup
        add     hl,de
        xor     a
        ld      (hl),a
        inc     hl
        ld      (hl),a
        ld      hl,linkdw
        add     hl,de
        ld      (hl),a
        inc     hl
        ld      (hl),a
        ret

demon:  ld      hl,linkdw
        add     hl,bc
        push    bc
        ld      c,(hl)
        inc     hl
        ld      b,(hl)
        ld      a,c
        or      b
        jp      z,nullend

        @acterdemon
nullend:
        pop     bc
        ret

link:   ld      hl,linkup
        add     hl,bc
        ld      (hl),e
        inc     hl
        ld      (hl),d

        ex      de,hl
        ld      e,(hl)
        ld      (hl),c
        inc     hl
        ld      d,(hl)
        ld      (hl),b

        ld      hl,linkdw
        add     hl,bc
        ld      (hl),e
        inc     hl
        ld      (hl),d

        ld      a,e
        or      d
        jp      z,qrtlink

        ld      hl,linkup
        add     hl,de
        ld      (hl),c
        inc     hl
        ld      (hl),b
qrtlink:
        ret

freeobj:
        ld      hl,linkdw
        add     hl,bc
        ld      e,(hl)
        inc     hl
        ld      d,(hl)
        push    de

        ld      hl,linkup
        add     hl,bc
        ld      e,(hl)
        inc     hl
        ld      d,(hl)
        ex      de,hl
        push    hl

        ld      e,(hl)
        inc     hl
        ld      d,(hl)
        ex      de,hl
        and     a
        sbc     hl,bc
        pop     hl
        jp      z,ltop

        ld      de,linkdw
        add     hl,de
ltop:
        pop     de
        ld      (hl),e
        inc     hl
        ld      (hl),d

        ld      a,e
        or      d
        jp      z,qrtfreeobj

        ld      hl,linkup
        add     hl,de
        push    hl

        ld      hl,linkup
        add     hl,bc
        ld      e,(hl)
        inc     hl
        ld      d,(hl)

        pop     hl
        ld      (hl),e
        inc     hl
        ld      (hl),d
qrtfreeobj:
        @superfreeobj
        ret

nextobj:
        ld      hl,linkdw
        add     hl,bc
        ld      e,(hl)
        inc     hl
        ld      d,(hl)
        ret
        end
```

リスト1-6-a holder.def

```
================== holder.def ================
;       meta class
metaclass:
        dw      metaclass
        dw      classMethod

;       class var

mclass  defl    0
imethod defl    2
memsiz  defl    4
cvarsize        equ     6

_selfnew        macro
        selfclass       0
        endm
_selfalloc      macro
        selfclass       2
        endm

_supernew       macro
        call    @@0011##
        endm
_superalloc     macro
        call    @@0000##
        endm
_acternew       macro
        callclass       0
        endm
_acteralloc     macro
        callclass       2
        endm
classMethod:
@@0016  equ     new
        public  @@0016
        dw      @@0016
        dw      @@0000##

;       instance var

class   defl    0
linkup  defl    2
linkdw  defl    4
holdhk  defl    6
ivarsize        equ     8

@selffreeobj    macro
        selfinstance    0
        endm
@selffree       macro
        selfinstance    2
        endm
@selfnextobj    macro
        selfinstance    4
        endm
@selflink       macro
        selfinstance    6
        endm
@selfdemon      macro
        selfinstance    8
        endm

@superfreeobj   macro
        call    @@0013##
        endm
@superfree      macro
        call    @@0002##
        endm
@supernextobj   macro
        call    @@0015##
        endm
@superlink      macro
        call    @@0014##
        endm
@superdemon     macro
        call    @@0012##
        endm
@acterfreeobj   macro
        callinstance    0
        endm
@acterfree      macro
        callinstance    2
        endm
@acternextobj   macro
        callinstance    4
        endm
@acterlink      macro
        callinstance    6
        endm
@acterdemon     macro
        callinstance    8
        endm
instanceMethod:
@@0018  equ     freeobj
        public  @@0018
@@0017  equ     demon
        public  @@0017
        dw      @@0018
        dw      @@0002##
        dw      @@0015##
        dw      @@0014##
        dw      @@0017
```

リスト1-6-b holder.mac

```
================== holder.mac ================
        include CLASS.DEF
        include HOLDER.DEF
holder::dw      metaclass
        dw      instancemethod
        dw      ivarsize

new:    _supernew
        ld      hl,holdhk
        add     hl,de
        xor     a
        ld      (hl),a
        inc     hl
        ld      (hl),a
        ret

demon:  ld      hl,holdhk
        add     hl,bc
        push    bc
        ld      c,(hl)
        inc     hl
        ld      b,(hl)
        ld      a,c
        or      b
        jp      z,nullend

        @acterdemon
nullend:
        pop     bc
        @superdemon
        ret

freeobj:
        ld      hl,holdhk
        add     hl,bc
        push    bc
        ld      c,(hl)
        inc     hl
        ld      b,(hl)
freeloop:
        ld      a,c
        or      b
        jp      z,qrtfree

        @acternextobj
        push    de
        @acterfreeobj
        pop     bc
        jp      freeloop
qrtfree:
        pop     bc
        @superfreeobj
        ret
        end
```

リスト1-7-a acheck.def

```
================== acheck.def ================
;       meta class
metaclass:
        dw      metaclass
        dw      classMethod

;       class var

mclass  defl    0
imethod defl    2
memsiz  defl    4
cvarsize        equ     6

_selfnew        macro
        selfclass       0
        endm
_selfalloc      macro
        selfclass       2
        endm

_supernew       macro
        call    @@0016##
        endm
_superalloc     macro
        call    @@0000##
        endm
_achecknew      macro
        callclass       0
        endm
_acheckalloc    macro
        callclass       2
        endm
classMethod:
        dw      @@0016##
        dw      @@0000##

;       instance var

class   defl    0
linkup  defl    2
linkdw  defl    4
holdhk  defl    6
ivarsize        equ     8

@selffreeobj    macro
        selfinstance    0
```

▲SHIFT BREAKを読んだが，U氏の問いに2つしか答えられない自分が悲しい。変身忍者嵐の愛馬はハヤブサオーだ。バトルホークの必殺技は，戦刃旋風斬り，変幻風刃投げ秘投三方刃である。
斎藤　正　(18)　埼玉県

```
        endm
@selffree       macro
        selfinstance    2
        endm
@selfnextobj    macro
        selfinstance    4
        endm
@selflink       macro
        selfinstance    6
        endm
@selfdemon      macro
        selfinstance    8
        endm
@selfatcheck    macro
        selfinstance    10
        endm

@superfreeobj   macro
        call    @@0018##
        endm
@superfree      macro
        call    @@0002##
        endm
@supernextobj   macro
        call    @@0015##
        endm
@superlink      macro
        call    @@0014##
        endm
@superdemon     macro
        call    @@0017##
        endm
@acheckfreeobj  macro
        callinstance    0
        endm
@acheckfree     macro
        callinstance    2
        endm
@achecknextobj  macro
        callinstance    4
        endm
@achecklink     macro
        callinstance    6
        endm
@acheckdemon    macro
        callinstance    8
        endm
@acheckatcheck  macro
        callinstance    10
        endm
instanceMethod:
@@0019  equ     atcheck
        public  @@0019
        dw      @@0018##
        dw      @@0002##
        dw      @@0015##
        dw      @@0014##
        dw      @@0017##
        dw      @@0019
```

リスト1-7-b acheck.mac

```
================== acheck.mac ================
        include CLASS.DEF
        include ACHECK.DEF
        dseg
atobj:: ds      2
atflag::ds      1
        cseg
acheck::dw      metaclass
        dw      instancemethod
        dw      ivarsize

atcheck:
        ld      hl,holdhk
        add     hl,bc
        push    bc
        ld      c,(hl)
        inc     hl
        ld      b,(hl)
tloop:
        ld      a,c
        or      b
        jp      z,qrtatcheck

        @achecknextobj
        push    de
        @acheckatcheck
        pop     bc
        jp      tloop
qrtatcheck:
        pop     bc
        ret
        end

```

リスト1-8-a backg.def

```
================= backg.def ===============
;       meta class
metaclass:
        dw      metaclass
        dw      classMethod

;       class var

mclass  defl    0
imethod defl    2
memsiz  defl    4
pnumb   defl    6
cvarsize        equ     7

_selfnew        macro
        selfclass       0
        endm
_selfalloc      macro
        selfclass       2
        endm

_supernew       macro
        call    @@0016##
        endm
_superalloc     macro
        call    @@0000##
        endm
_sheadnew       macro
        callclass       0
        endm
_sheadalloc     macro
        callclass       2
        endm
_sheadmakep     macro
        callclass       4
        endm
_screennew      macro
        callclass       0
        endm
_screenalloc    macro
        callclass       2
        endm
classMethod:
@@0031  equ     new
        public  @@0031
@@0030  equ     alloc
        public  @@0030
        dw      @@0031
        dw      @@0030

;       instance var

class   defl    0
linkup  defl    2
linkdw  defl    4
holdhk  defl    6
setp    defl    8
xpoint  defl    9
ypoint  defl    10
ivarsize        equ     11

@selffreeobj    macro
        selfinstance    0
        endm
@selffree       macro
        selfinstance    2
        endm
@selfnextobj    macro
        selfinstance    4
        endm
@selflink       macro
        selfinstance    6
        endm
@selfdemon      macro
        selfinstance    8
        endm
@selfatcheck    macro
        selfinstance    10
        endm

@superfreeobj   macro
        call    @@0018##
        endm
@superfree      macro
        call    @@0002##
        endm
@supernextobj   macro
        call    @@0015##
        endm
@superlink      macro
        call    @@0014##
        endm
@superdemon     macro
        call    @@0017##
        endm
@superatcheck   macro
        call    @@0019##
        endm
@sheadfreeobj   macro
        callinstance    0
        endm
@sheadfree      macro
        callinstance    2
        endm
@sheadnextobj   macro
        callinstance    4
        endm
@sheadlink      macro
        callinstance    6
        endm
@sheaddemon     macro
        callinstance    8
        endm
@sheadatcheck   macro
        callinstance    10
        endm
@sheadclears    macro
        callinstance    12
        endm
@sheadshows     macro
        callinstance    14
        endm
@screenfreeobj  macro
        callinstance    0
        endm
@screenfree     macro
        callinstance    2
        endm
@screenbox      macro
        callinstance    4
        endm
@screenputc     macro
        callinstance    6
        endm
instanceMethod:
@@0034  equ     atcheck
        public  @@0034
@@0033  equ     free
        public  @@0033
@@0032  equ     demon
        public  @@0032
        dw      @@0018##
        dw      @@0033
        dw      @@0015##
        dw      @@0014##
        dw      @@0032
        dw      @@0034
```

リスト1-8-b backg.mac

```
================== backg.mac =================
        include CLASS.DEF
        include BACKG.DEF
LEFT    EQU     01
RIGHT   EQU     39
TOP     EQU     01
BOTOM   EQU     24
        dseg
bwork:: ds      ivarsize
        cseg
backg:: dw      metaclass
        dw      instancemethod
        dw      ivarsize
        db      1

alloc:  ld      de,bwork
        ret

new:    _supernew
        push    de
        push    bc
        push    bc
        ld      bc,swork##
        @screenbox
        pop     bc
        ld      hl,pnumb
        add     hl,bc
        ld      a,(hl)
newloop:
        push    af
        ld      (un##),a
        ld      bc,shead##
        _sheadmakep
        ld      b,d
        ld      c,e
        ld      de,bwork+holdhk
        @sheadlink
        pop     af
        and     a
        jp      z,qrtnew
        dec     a
        jp      newloop
qrtnew:
        pop     bc
        pop     de
        ret

demon:  ld      hl,20000
dloop:  dec     hl
        ld      a,l
        or      h
        jp      nz,dloop

        @superdemon
        ld      hl,holdhk
        add     hl,bc
        ld      e,(hl)
        inc     hl
        ld      d,(hl)
        ld      a,e
        or      d
        jp      nz,demon
        ret

atcheck:
        ld      de,(atobj##)
        ld      hl,xpoint
        add     hl,de
        ld      a,(hl)
        cp      LEFT
        jp      m,aton

        cp      RIGHT
        jp      p,aton

        ld      hl,ypoint
        add     hl,de
        ld      a,(hl)
        cp      TOP
        jp      m,aton

        cp      BOTOM
        jp      p,aton
        @superatcheck
        ret

aton:   ld      a,0ffh
        ld      (atflag##),a
        ret

free:   ret
        end
```

リスト1-9-a snakep.def

```
================== snakep.def ===============
;       meta class
metaclass:
        dw      metaclass
        dw      classMethod

;       class var

mclass  defl    0
imethod defl    2
memsiz  defl    4
cvarsize        equ     6

_selfnew        macro
        selfclass       0
        endm
_selfalloc      macro
        selfclass       2
        endm
_selfmakep      macro
        selfclass       4
        endm

_supernew       macro
        call    @@0016##
        endm
```

```
_superalloc     macro
        call    @@0000##
        endm
_screennew      macro
        callclass       0
        endm
_screenalloc    macro
        callclass       2
        endm
classMethod:
@@0020  equ     makep
        public  @@0020
        dw      @@0016##
        dw      @@0000##
        dw      @@0020

;       instance var

class   defl    0
linkup  defl    2
linkdw  defl    4
holdhk  defl    6
setp    defl    8
xpoint  defl    9
ypoint  defl    10
ivarsize        equ     11

@selffreeobj    macro
        selfinstance    0
        endm
@selffree       macro
        selfinstance    2
        endm
@selfnextobj    macro
        selfinstance    4
        endm
@selflink       macro
        selfinstance    6
        endm
@selfdemon      macro
        selfinstance    8
        endm
@selfatcheck    macro
        selfinstance    10
        endm
@selfclears     macro
        selfinstance    12
        endm
@selfshows      macro
        selfinstance    14
        endm

@superfreeobj   macro
        call    @@0018##
        endm
@superfree      macro
        call    @@0002##
        endm
@supernextobj   macro
        call    @@0015##
        endm
@superlink      macro
        call    @@0014##
        endm
@superdemon     macro
        call    @@0017##
        endm
@superatcheck   macro
        call    @@0019##
        endm
@screenfreeobj  macro
        callinstance    0
        endm
@screenfree     macro
        callinstance    2
        endm
@screenbox      macro
        callinstance    4
        endm
@screenputc     macro
        callinstance    6
        endm
instanceMethod:
@@0024  equ     freeobj
        public  @@0024
@@0023  equ     clears
        public  @@0023
@@0022  equ     shows
        public  @@0022
@@0021  equ     atcheck
        public  @@0021
        dw      @@0024
        dw      @@0002##
        dw      @@0015##
        dw      @@0014##
        dw      @@0017##
        dw      @@0021
        dw      @@0023
        dw      @@0022
```

リスト1-9-b snakep.mac

```
================== snakep.mac ================
        include CLASS.DEF
        include SNAKEP.DEF
        dseg
xa::    ds      1
ya::    ds      1
        cseg
snakep::dw      metaclass
        dw      instancemethod
        dw      ivarsize

makep:  _selfnew
        ld      hl,xpoint
        add     hl,de
        ld      a,(xa)
        ld      (hl),a
        ld      hl,ypoint
        add     hl,de
        ld      a,(ya)
        ld      (hl),a
        ret

atcheck:
        ld      hl,(atobj##)
        and     a
        sbc     hl,bc
        jp      z,qrtatcheck

        ld      de,(atobj##)
        ld      hl,xpoint
        add     hl,de
        ld      a,(hl)
        ld      hl,xpoint
        add     hl,bc
        cp      (hl)
        jp      nz,qrtatcheck

        ld      hl,ypoint
        add     hl,de
        ld      a,(hl)
        ld      hl,ypoint
        add     hl,bc
        cp      (hl)
        jp      nz,qrtatcheck

        ld      hl,setp
        add     hl,bc
        ld      a,(hl)
        ld      (atflag##),a
qrtatcheck:
        @superatcheck
        ret

shows:  ld      hl,xpoint
        add     hl,bc
        ld      a,(hl)
        ld      (xp##),a
        ld      hl,ypoint
        add     hl,bc
        ld      a,(hl)
        ld      (yp##),a
        ld      hl,setp
        add     hl,bc
        ld      a,(hl)
        ld      (pat##),a
        push    bc
        ld      bc,swork##
        @screenputc
        pop     bc
        ret

clears: ld      hl,xpoint
        add     hl,bc
        ld      a,(hl)
        ld      (xp##),a
        ld      hl,ypoint
        add     hl,bc
        ld      a,(hl)
        ld      (yp##),a
        ld      a,' '
        ld      (pat##),a
        push    bc
        ld      bc,swork##
        @screenputc
        pop     bc
        ret

freeobj:
        @selfclears
        @superfreeobj
        ret
        end
```

リスト1-10-a sbody.def

```
================== sbody.def ================
;       meta class
metaclass:
        dw      metaclass
        dw      classMethod

;       class var

mclass  defl    0
imethod defl    2
memsiz  defl    4
charc   defl    6
cvarsize        equ     7

_selfnew        macro
        selfclass       0
        endm
_selfalloc      macro
        selfclass       2
        endm
_selfmakep      macro
        selfclass       4
        endm

_supernew       macro
        call    @@0016##
        endm
_superalloc     macro
        call    @@0000##
        endm
_supermakep     macro
        call    @@0020##
        endm
classMethod:
@@0025  equ     makep
        public  @@0025
        dw      @@0016##
        dw      @@0000##
        dw      @@0025

;       instance var

class   defl    0
linkup  defl    2
linkdw  defl    4
holdhk  defl    6
setp    defl    8
xpoint  defl    9
ypoint  defl    10
tcount  defl    11
ivarsize        equ     13

@selffreeobj    macro
        selfinstance    0
        endm
@selffree       macro
        selfinstance    2
        endm
@selfnextobj    macro
        selfinstance    4
        endm
@selflink       macro
        selfinstance    6
        endm
@selfdemon      macro
        selfinstance    8
        endm
@selfatcheck    macro
        selfinstance    10
        endm
@selfclears     macro
        selfinstance    12
        endm
@selfshows      macro
        selfinstance    14
        endm

@superfreeobj   macro
        call    @@0024##
        endm
@superfree      macro
        call    @@0002##
        endm
@supernextobj   macro
        call    @@0015##
        endm
@superlink      macro
        call    @@0014##
        endm
@superdemon     macro
        call    @@0017##
        endm
@superatcheck   macro
        call    @@0021##
        endm
@superclears    macro
        call    @@0023##
        endm
@supershows     macro
        call    @@0022##
        endm
instanceMethod:
@@0026  equ     demon
        public  @@0026
        dw      @@0024##
        dw      @@0002##
        dw      @@0015##
        dw      @@0014##
        dw      @@0026
        dw      @@0021##
        dw      @@0023##
        dw      @@0022##
```

リスト1-10-b sbody.mac

```
================== sbody.mac ================
        include CLASS.DEF
        include SBODY.DEF
        dseg
ca::    ds      2
bcc::   ds      1
        cseg
sbody:: dw      metaclass
        dw      instancemethod
        dw      ivarsize

makep:  _supermakep
        push    bc
        push    de
        ld      a,(bcc)
        ld      hl,setp
        add     hl,de
        ld      (hl),a
        ld      b,d
        ld      c,e
        @selfshows
        pop     de
        pop     bc
        ld      hl,tcount
        add     hl,de
        push    de
        ld      de,(ca)
        ld      (hl),e
        inc     hl
        ld      (hl),d
        pop     de
        ret

demon:  @superdemon
        ld      hl,tcount
        add     hl,bc
        ld      e,(hl)
        inc     hl
        ld      d,(hl)
        dec     de
        ld      (hl),d
        dec     hl
        ld      (hl),e
        ld      a,e
        or      d
        jp      nz,qrtdemon

        @selffreeobj
qrtdemon:
        ret
        end
```

▲X1Cを買ってからすぐにディスクの時代に入ってしまい，不満の続く日々でしたが，X68000 ACE-HDに感動してこのたび購入を決意しました。Oh!MZともしばらく別れていましたが，また毎月読もうと思います。おっと，もうOh!Xでしたっけね。これからもよろしく。

倉橋 雅夫 (19) 京都府

リスト1-11-a shead.def

```
================== shead.def ================
;           meta class
metaclass:
            dw      metaclass
            dw      classMethod

;           class var

mclass  defl    0
imethod defl    2
memsiz  defl    4
cvarsize        equ     6

_selfnew        macro
            selfclass       0
            endm
_selfalloc      macro
            selfclass       2
            endm
_selfmakep      macro
            selfclass       4
            endm

_supernew       macro
            call    @@0016##
            endm
_superalloc     macro
            call    @@0000##
            endm
_supermakep     macro
            call    @@0020##
            endm
_acheckne w     macro
            callclass       0
            endm
_acheckalloc    macro
            callclass       2
            endm
_joypotnew      macro
            callclass       0
            endm
_joypotalloc    macro
            callclass       2
            endm
_sbodynew       macro
            callclass       0
            endm
_sbodyalloc     macro
            callclass       2
            endm
_sbodymakep     macro
            callclass       4
            endm
classMethod:
@@0028  equ     new
            public  @@0028
@@0027  equ     makep
            public  @@0027
            dw      @@0028
            dw      @@0000##
            dw      @@0027

;           instance var

class   defl    0
linkup  defl    2
linkdw  defl    4
holdhk  defl    6
setp    defl    8
xpoint  defl    9
ypoint  defl    10
unumb   defl    11
mcount  defl    12
gcount  defl    14
xmove   defl    16
ymove   defl    17
blp     defl    18
ivarsize        equ     19

@selffreeobj    macro
            selfinstance    0
            endm
@selffree       macro
            selfinstance    2
            endm
@selfnextobj    macro
            selfinstance    4
            endm
@selflink       macro
            selfinstance    6
            endm
@selfdemon      macro
            selfinstance    8
            endm
@selfatcheck    macro
            selfinstance    10
            endm
@selfclears     macro
            selfinstance    12
            endm
@selfshows      macro
            selfinstance    14
            endm

@superfreeobj   macro
            call    @@0024##
            endm
@superfree      macro
            call    @@0002##
            endm
@supernextobj   macro
            call    @@0015##
            endm
@superlink      macro
            call    @@0014##
            endm
@superdemon     macro
            call    @@0017##
            endm
@superatcheck   macro
            call    @@0021##
            endm
@superclears    macro
            call    @@0023##
            endm
@supershows     macro
            call    @@0022##
            endm
@acheckfreeobj  macro
            callinstance    0
            endm
@acheckfree     macro
            callinstance    2
            endm
@achecknextobj  macro
            callinstance    4
            endm
@achecklink     macro
            callinstance    6
            endm
@acheckdemon    macro
            callinstance    8
            endm
@acheckatcheck  macro
            callinstance    10
            endm
@joypotfreeobj  macro
            callinstance    0
            endm
@joypotfree     macro
            callinstance    2
            endm
@joypotinp      macro
            callinstance    4
            endm
@sbodyfreeobj   macro
            callinstance    0
            endm
@sbodyfree      macro
            callinstance    2
            endm
@sbodynextobj   macro
            callinstance    4
            endm
@sbodylink      macro
            callinstance    6
            endm
@sbodydemon     macro
            callinstance    8
            endm
@sbodyatcheck   macro
            callinstance    10
            endm
@sbodyclears    macro
            callinstance    12
            endm
@sbodyshows     macro
            callinstance    14
            endm
instanceMethod:
@@0029  equ     demon
            public  @@0029
            dw      @@0024##
            dw      @@0002##
            dw      @@0015##
            dw      @@0014##
            dw      @@0029
            dw      @@0021##
            dw      @@0023##
            dw      @@0022##
```

リスト1-11-b shead.mac

```
================== shead.mac ================
            include CLASS.DEF
            include SHEAD.DEF
COUNTD  equ     30
ADDTIME equ     60
ADDVAL  equ     10
            dseg
un::    ds      1
xm:     ds      1
ym:     ds      1
uc:     ds      1
bkc:    ds      1
            cseg
shead:: dw      metaclass
            dw      instancemethod
            dw      ivarsize

new:    _supernew
            ld      hl,gcount
            add     hl,de
            push    de
            ld      de,COUNTD
            ld      (hl),e
            inc     hl
            ld      (hl),d
            pop     de

            ld      hl,mcount
            add     hl,de
            push    de
            ld      de,ADDTIME
            ld      (hl),e
            inc     hl
            ld      (hl),d
            pop     de

            xor     a
            ld      hl,xmove
            add     hl,de
            ld      (hl),a
            ld      hl,ymove
            add     hl,de
            ld      (hl),a
            ret

makep:  ld      a,(un)
            ld      l,a
            ld      h,0
            add     hl,hl
            ld      de,pltable
            add     hl,de
            ld      e,(hl)
            inc     hl
            ld      d,(hl)
            ex      de,hl
            ld      a,(hl)
            ld      (xa##),a
            inc     hl
            ld      a,(hl)
            ld      (ya##),a
            inc     hl
            ld      a,(hl)
            ld      (xm),a
            inc     hl
            ld      a,(hl)
            ld      (ym),a
            inc     hl
            ld      a,(hl)
            ld      (uc),a
            inc     hl
            ld      a,(hl)
            ld      (bkc),a
            _supermakep
            ld      a,(un)
            ld      hl,unumb
            add     hl,de
            ld      (hl),a
            ld      a,(xm)
            ld      hl,xmove
            add     hl,de
            ld      (hl),a
            ld      a,(ym)
            ld      hl,ymove
            add     hl,de
            ld      (hl),a
            ld      a,(uc)
            ld      hl,setp
            add     hl,de
            ld      (hl),a
            ld      a,(bkc)
            ld      hl,blp
            add     hl,de
            ld      (hl),a
            ret

demon:  @superdemon
            ld      hl,mcount
            add     hl,bc
            ld      e,(hl)
            inc     hl
            ld      d,(hl)
            dec     de
            ld      (hl),d
            dec     hl
            ld      (hl),e

            ld      a,e
            or      d
            jp      nz,sctadd

            ld      de,ADDTIME
            ld      (hl),e
            inc     hl
            ld      (hl),d

            ld      hl,gcount
            add     hl,bc
            ld      e,(hl)
            inc     hl
            ld      d,(hl)
            push    hl
            ld      hl,ADDVAL
            add     hl,de
            ex      de,hl
            pop     hl
            ld      (hl),d
            dec     hl
            ld      (hl),e

sctadd: ld      hl,gcount
            add     hl,bc
            ld      e,(hl)
            inc     hl
            ld      d,(hl)
            ld      (ca##),de

            ld      hl,xpoint
            add     hl,bc
            ld      a,(hl)
            ld      (xa##),a
            ld      hl,ypoint
            add     hl,bc
            ld      a,(hl)
            ld      (ya##),a
            ld      hl,blp
            add     hl,bc
            ld      a,(hl)
            ld      (bcc##),a
            push    bc
            ld      bc,sbody##
            _sbodymakep
            ld      b,d
            ld      c,e
            ld      de,bwork##+holdhk
            @sbodylink
            pop     bc

            ld      hl,unumb
            add     hl,bc
            ld      a,(hl)
            push    bc
            ld      bc,jwork##
            @joypotinp
            pop     bc
            or      11110000b
            inc     a
            jp      z,notinp

            dec     a
            ld      d,4
            ld      hl,joytable
joylop:
            rrca
            jp      nc,readj
            inc     hl
            inc     hl
```

▲Oh!Xの読者のなかにも中島みゆきファンはいる(と，6月号133ページの榎木さんにお伝えください)。
武田 泰典 (19) 茨城県

```
        dec     d
        jp      nz,joylop
        jp      notinp
readj:
        ld      a,(hl)
        push    hl
        ld      hl,xmove
        add     hl,bc
        ld      (hl),a
        pop     hl
        inc     hl
        ld      a,(hl)
        ld      hl,ymove
        add     hl,bc
        ld      (hl),a
notinp:
        ld      hl,xmove
        add     hl,bc
        ld      a,(hl)
        ld      hl,xpoint
        add     hl,bc
        add     a,(hl)
        ld      (hl),a

        ld      hl,ymove
        add     hl,bc
        ld      a,(hl)
        ld      hl,ypoint
        add     hl,bc
        add     a,(hl)
        ld      (hl),a
        @selfshows

        xor     a
        ld      (atflag##),a
        ld      (atobj##),bc
        push    bc
        ld      bc,bwork##
        @acheckatcheck
        pop     bc
        ld      a,(atflag##)
        or      a
        ret     z

        @selffreeobj
        ret

joytable:
        db      00,-1,00,01,-1,00,1,00

pltable:
        dw      pl0,pl1,pl2,pl3

pl0:    db      10,13,0,-1
        db      '@','0'
pl1:    db      30,12,0,1
        db      '*','1'
pl2:    db      20,6,-1,0
        db      '$','2'
pl3:    db      20,19,1,0
        db      '%','3'
        end
```

リスト1-12-a joypot.def

```
=================  joypot.def  ================
;       meta class
metaclass:
        dw      metaclass
        dw      classMethod

;       class var

mclass  defl    0
imethod defl    2
memsiz  defl    4
cvarsize        equ     6

_selfnew        macro
        selfclass       0
        endm
_selfalloc      macro
        selfclass       2
        endm

_supernew       macro
        call    @@0001##
        endm
_superalloc     macro
        call    @@0000##
        endm
classMethod:
@@0008  equ     alloc
        public  @@0008
        dw      @@0001##
        dw      @@0008

;       instance var

class   defl    0
ivarsize        equ     2

@selffreeobj    macro
        selfinstance    0
        endm
@selffree       macro
        selfinstance    2
        endm
@selfinp        macro
        selfinstance    4
        endm

@superfreeobj   macro
        call    @@0003##
        endm
@superfree      macro
        call    @@0002##
        endm
instanceMethod:
@@0010  equ     free
        public  @@0010
@@0009  equ     inp
        public  @@0009
        dw      @@0003##
        dw      @@0010
        dw      @@0009
```

リスト1-12-b joypot.mac

```
=================  joypot.mac  ================
        include CLASS.DEF
        include JOYPOT.DEF
        dseg
jwork:: ds      ivarsize
        cseg
joypot::dw      metaclass
        dw      instancemethod
        dw      ivarsize

alloc:
        ld      de,jwork
        ret

inp:    push    af
        ld      bc,1c00h
        ld      a,07
        out     (c),a
        ld      bc,1b00h
        xor     a
        out     (c),a
        pop     af
        add     a,14
        ld      bc,1c00h
        out     (c),a
        ld      bc,1b00h
        in      a,(c)
        ret

free:   ret
        end
```

リスト1-13-a screen.def

```
=================  screen.def  ================
;       meta class
metaclass:
        dw      metaclass
        dw      classMethod

;       class var

mclass  defl    0
imethod defl    2
memsiz  defl    4
cvarsize        equ     6

_selfnew        macro
        selfclass       0
        endm
_selfalloc      macro
        selfclass       2
        endm

_supernew       macro
        call    @@0001##
        endm
_superalloc     macro
        call    @@0000##
        endm
classMethod:
@@0004  equ     alloc
        public  @@0004
        dw      @@0001##
        dw      @@0004

;       instance var

class   defl    0
ivarsize        equ     2

@selffreeobj    macro
        selfinstance    0
        endm
@selffree       macro
        selfinstance    2
        endm
@selfbox        macro
        selfinstance    4
        endm
@selfputc       macro
        selfinstance    6
        endm

@superfreeobj   macro
        call    @@0003##
        endm
@superfree      macro
        call    @@0002##
        endm
instanceMethod:
@@0007  equ     box
        public  @@0007
@@0006  equ     free
        public  @@0006
@@0005  equ     putc
        public  @@0005
        dw      @@0003##
        dw      @@0006
        dw      @@0007
        dw      @@0005
```

リスト1-13-b screen.mac

```
=================  screen.mac  ================
        include CLASS.DEF
        include SCREEN.DEF
        dseg
swork:: ds      ivarsize
xp::    ds      1
yp::    ds      1
pat::   ds      1
        cseg
screen::dw      metaclass
        dw      instancemethod
        dw      ivarsize

alloc:
        ld      de,swork
        ret

putc:
        push    bc
        ld      a,(yp)
        ld      c,a
        ld      l,a
        ld      b,00
        ld      h,00
        add     hl,hl
        add     hl,hl
        add     hl,bc
        add     hl,hl
        add     hl,hl
        add     hl,hl
        ld      a,(xp)
        ld      c,a
        ld      b,00
        add     hl,bc
        ld      bc,3000h
        add     hl,bc
        ld      c,l
        ld      b,h
        ld      a,(pat)
        out     (c),a
        pop     bc
        ret

box:
        push    bc
        ld      bc,2000h
        ld      hl,1000h
        ld      d,07h
        call    clear
        ld      bc,3000h
        ld      hl,1000h
        ld      d,' '
        call    clear
        ld      a,'#'
        ld      (pat),a
        ld      a,00
        ld      (yp),a
        call    hline
        ld      a,24
        ld      (yp),a
        call    hline
        ld      a,00
        ld      (xp),a
        call    vline
        ld      a,39
        ld      (xp),a
        call    vline
        pop     bc
        ret

clear:  out     (c),d
        inc     bc
        dec     hl
        ld      a,l
        or      h
        jp      nz,clear
        ret

hline:
        ld      a,39
        ld      (xp),a
hloop:
        call    putc
        ld      a,(xp)
        and     a
        ret     z

        dec     a
        ld      (xp),a
        jp      hloop

vline:
        ld      a,24
        ld      (yp),a
vloop:
        call    putc
        ld      a,(yp)
        and     a
        ret     z

        dec     a
        ld      (yp),a
        jp      vloop


free:   ret
        end
```

リスト2 リスト1をリンクするバッチファイル

```
180 /d:800,runtime,start,object,acter,holder,acheck,backg,snakep,sbody,shead,joy
pot,screen,s/n/y/e
```

▲「あぶない福袋」は虚実入り乱れているようで本当にあぶない。もっとあぶないのは突如現れた満開製作所の広告だ。「あぶない福袋」の子ディレクトリのような気もしたが，即日6,000円持って郵便局へ走った私であった。 高橋 良和 (34) 東京都

危険な事情

Iwai Ippei 祝　一平

大地最低の作戦

しばらく前，テレビや新聞をにぎわしたニュースに，「法律を変えて，土地の所有権を地下50メートルまでに制限する」というのがあった。

これはよーするに，政府の無能によって土地の値段がガバガバになってしまったので，地下鉄の建設費が巨額になったことへの対策であるらしい。現在の法律では地下鉄を通すとき，土地代の数割を支払うことになっているのだそうだ。

そいでもって，私はこのニュースを聞いたとたん，

地下帝国を建設する

という，マンダムな野望を思いついたのであった。土地の権利が地下50メートルまでに制限されるということは，よーするにそれより下は誰のものでもないということではないか。つまりは**早い者勝ち**ということじゃないか。であるからして，とりあえずはなんとかして10坪ぐらいの土地を買い，下に50メートル掘り進む。そして法律が成立するのをコシタンタンと待ち，一目散に横にガンガンと掘り進む。これで地下帝国のいっちょ上がりである。こうすれば，日当たりは悪いが，エレベータに乗って1分で都心に出られるという最高の場所がしこたま手に入るのである。下に掘る分は OK なのだから，地下50メートルから始まる“低層建築”で巨大なビルを建てれば，たちまち大金持ちである。

てなことをホイホイと夢想していたのだが，落ち着いて考え直してみると，これだけオイシイのであれば，やっぱり国有ということになってしまって，利権とか使用権とかは，ぜーんぶ政治家の手の中という，いつもの結末が見えてしまった。そうだよなー。さもないと，東京の地下が穴だらけのボコボコになって，そこらじゅうで落盤してしまうのである。

しかし待てよ。たとえば新宿や渋谷や丸の内の地下50メートルに駅を作れるとなると，これはとんでもない利権ではないか。いやいや，別に地下鉄の駅と決まったわけではない。原則的に国有地になるんだから，払い下げを受けて地下街を作るということだってあり得るじゃないか。

この方式による地下鉄建設には，営団地下鉄などの公共団体だけではなく，西武グループなんかも意欲を示しているそーである。そうなると，西武鉄道がねっとりと都心に乗り込んでくるわけだ，そいでもって，地下に大デパートができて……。

そうである。もしかしたら，この「地下50メートル」という話には，ものすごく“危険な事情”が秘められているのではないだろうか。だって，都心の一等地ばかりではなく，日本全土がどどーんと“国有地”になるのである。うんと安く払い下げるとしても数兆円は下らないであろう。これはとんでもない利権ではないか。うう～ん，私も選挙に立候補したくなってきた。

しかし待てよ。もしも勝手に地下を国有地にされてしまったら，今までは当然の権利であったはずの，温泉を掘るためのボーリング工事とか，高いビルを作るための基礎工事とかができなくなるではないか。これはおかしいぞ。よし，もしも私の土地(持ってないけど）の下に勝手に地下鉄なんぞを作ったら，線路の真ん中にボーリング用のパイプを打ち込んで，「温泉を掘るんだあー」といってダダをこねてやろう。

〽TRONのよーで，TRONじゃない，ベンベン

最近あちこちでうごめいている「国定パソコン」であるが，私はこの教育パソコンにも危険な事情が潜んでいるのではないかとニラんでいる。試作機は8社から出されているが，実は全部松下と富士通の OEM なので，本当は2種類しかないそーである。ほーらほら，危険なにおいがしてきたであろう。

で，この教育パソコンのキーボードがち



ょっとすごい。どうすごいかというと，なんと新 JIS 規格なのである（現在のキーボードはたいていが旧JIS規格)。きっとほとんどの読者は，新 JIS のキーボードを見たことがないであろう。実は私も見たことがないのである。そこで，図1に教育パソコンなるもののキー配置を示しておく。

まず，注目していただきたいのが，右の[上段]キー（SHIFTのことだな）のすぐ左隣に，普通のパソコンには必ずあるはずの，[ロ]（と[_]：アンダーバー）がないということである。新 JIS ではそのように決められてしまっているのである。それから，かなの配列が現在のパソコンのキーボードとはぜんぜん違うというのも大胆である。特に奇怪なのが，母音などの大文字と小文字の配置である。現在のパソコンでは，かなモードの時に［Z］で“つ”，[SHIFT]＋［Z］で“っ”などとなっているのであるが，新 JIS では何を考えたのか，「あいうえおやゆよつ」と「ぁぃぅぇぉゃゅょっ」の位置関係がバラバラなのである。英字キーの位置で示すと，

大文字　→　B K J U j o + g Y
小文字　→　q a z x c b f T N

と，なっている。やっと国産パソコンのキーボードの配置が統一されつつあるというのに，なんで今さらこんなものを持ち出してきたのであろうか。JIS だからといって，こんな横暴が許されていいのか？

それからなんといっても，右端のテンキー部分でキラリと光っている［と］(小学校では加算記号“＋”を習う前に「と」を使う)，［分の］という2つのキーは見逃せないであろう。パソコンの常識で考えれば，

こういうものは無理やりキーにしたりせずに，ファンクションキーに割り当てるべきじゃないのか。ファンクションキーなら，ラベルを変えることが可能だろうし，キー定義を変えることだって可能だ。全学年で必要なものならまだしも，実際こんなキーは小学校専用じゃないか。教育パソコンは中学校でも使われる予定なんだろ？　だったら中学校ではまったくの無駄じゃないか。いいや無駄だけではすまない。むしろ邪魔ではないのか。どーしてファンクションキーに割り当てなかったんだ？　ファンクションキーとは，こういうときのためにあるんだぞ。何考えてんだ，あんたたち。

ほかにも［制御］（[CTRL］だな）の位置が狂ってるとか，スペースが異常に右に寄っているとか，もう勝手にしてくれといいたくなるぐらい斬新なキーボードである。これじゃ現在のパソコンのキーボードと非コンパチではないか。それも，なんの必然性もない変更ばかりだ。もはやこうなったら責任者出てこい，である。

そして肝心のソフトであるが，さすがにワープロやカルクなどをひと通り揃えなければ大ひんしゅくだということには気がついているようで，現在各社で手分けして作っている最中らしい。そして，移植を促進するために補助金も出るそうである。しかしどう考えても，結局ソフトの主流は98からの移植ということにならざるを得ないのではないだろうか？　つまり，98用だったらすでに存在しているものを，わざわざ移植することになる。そうなってくると，ハードウェアの性能はだいたい同じなのだから，どうやっても，98のソフトの質と量に追いつくのに，相当な時間がかかるのではないだろうか。

教育パソコンを98以外のものにしたいという気持ちはわからないわけじゃない。だけど，使われているのは税金なんだぞ。できたパソコンやソフトを小中学校が買うにしても，その金はぜーんぶ税金から出るんだぞ。総額９千億円なんだぞ。こんな短期間にわたわたとやって，ちゃんと世間様に顔向けできるマシンとソフトが作れるのか？

この教育パソコンに関しては，松下が一番乗り気であるらしい。ううむ，実にキナくさい。そういえば，数年前に松下がIBM-PCのコンパチマシンを米国に輸出したが，たちまちIBMからクレームがつき，あわてて引っ込めたという出来事があったな。あの背景は，実はIBM-PCのハードウェアを作っていたのは松下だったということによるものらしい。そのような危険な事情があるにもかかわらず，コンパチマシンに手を出したのであるから，当然といえば当然の結果だったのかもしれない。で，この危険な事情が今も尾を引いているので，松下は「AX」に手を出せないでいるのではないか。なぜならばAXはPC/ATとコンパチだからである。ここにMSXがいまいちだという事情も加わり，松下は教育パソコンに活路を求めているのではないだろうか。

そもそも教育に規格を持ち込むという発想からしてものすごい。

坂村のケンちゃんにしても，一世一代で頑張っているBTRONのOSを，このような中途半端な機械（失礼）に流用されるのは不本意なはずであろう。しかし，文句をいう気配がない。原著作権を持ってるんだから，「こんなパソコンのためにBTRONをやってるんじゃねぇ！」とかいって怒りそうなものであるが，そうしない。ふむふむ。きっと坂村氏には坂村氏なりの危険な事情があるのだろう。

ホットな島(だって熱帯だモン)

総費用300億円という沖ノ鳥島の保全工事が始まった。そのニュースを見てて思いついたことであるが（最近はニュースを見ながら，いろんなことを思いつく），もしも，こっそり船で沖ノ鳥島に行って，誰も見ていないうちに２つの岩を爆破しちまっていたらどーなったであろうか。工事に来た人たちはおろか，日本中が腰を抜かす空前絶後のプラクティカルジョークになったのではないかと思うのであるが，どーであろう。国際法では人工物は領土と認められないそうだから，あわててコンクリートかなんかでペタペタとくっつけ直してもだめなはずである。ということは，ダイナマイト数本で「日本全土とほぼ同じ広さの経済水域」が“ぱぁ”になってしまうわけだ。ううむ。やっぱりこれは，ちょっと笑えないか。

そのニュース報道によれば，戦前にも沖ノ鳥島で工事があったそうである。そのときは，気象観測のための施設を作るということだったが，実はそれはタテマエで，本当は，軍事施設を作りたかったんだそうである。

ということは，今回の「経済水域を確保する」という名目の裏にも，ひょっとしたら危険な事情が隠されているのかもしれない。公にはできない何か。そうだそうだ。現在の日本の土木技術と経済力があれば，レーダー基地ぐらいなら簡単に作れてしまうではないか。そう考えていくと，300億円という費用があれだけスンナリと決まったというのも説明できるではないか。う～ん，危険危険。

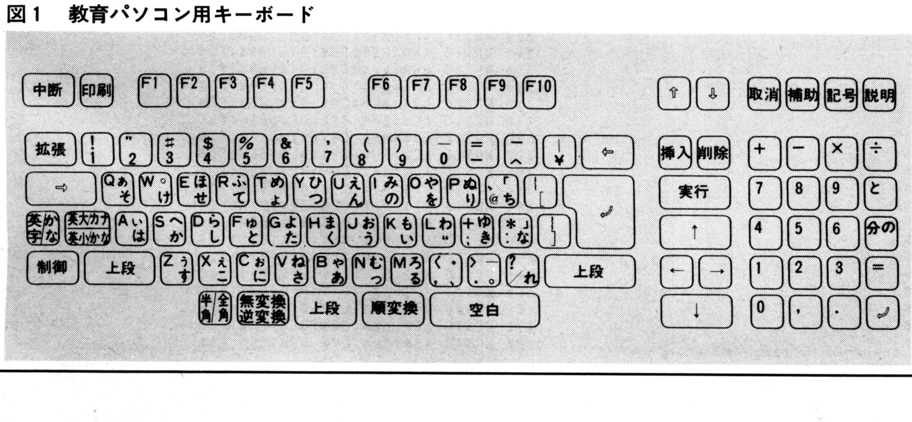
ああ，NHK

新聞に載ってたが，NHKの世論調査で，税制改革に反対する意見が急増し，全体の48％に達しているという結果が出たそうである。しかし，NHKはそれを放送せずに内部の関係雑誌に載せるにとどめたそうだ。NHKはその理由として，調査結果が出る直前に自民党の税調が答申を発表したので，それに対する反対だと誤解されるのを避けるためだといっているそーである。何考えてんだコノヤロ。ちゃんと，「これはいついつの調査であって，先日の何々とは関係ありません」と断りを付ければなんの問題もないだろう。受信料は非課税になるのかなー？　衛星放送の有料化は許可されるのかなー？　うむうむ。危険な事情。

♨　♨

そーゆーわけで突然最終回なのであった。どーして急に終わるのかつーと，やはりそこには危険な事情があったりするのである。ではC調言語で会いましょう，であった。

図１　教育パソコン用キーボード



MZ-1500用
テクノポリス/邂逅
X1/X1turbo用
アフターバーナー
MZ-2500用
TRUTH

Mori Hiroshi
森　弘

Kaneko Shunichi
金子　俊一

Kurata Yoshto
倉田　嘉人

今月は久びさにMZ-1500用の作品も登場です。曲目はちょっと懐かしのYMOの作品から2曲。X1用にはゲームセンターでお馴染みのアフターバーナー，そしてMZ-2500用にはSQUAREのTRUTHと，バラエティ豊富にお届けします。そろそろ，クラシックも欲しいところですが……。

久びさのMZ-1500用です

まずはMZ-1500のPSG6音によるYMOのTECHNOPOLISと邂逅をメドレーで聞いてもらいましょう。MZ-1500用の投稿はあまり数が多くないのですが，1つひとつのレベルは結構高いようですね。このプログラムではPSGでノイズを出すためにBASICを一部書き換えています。ドラムパートなどの処理で苦労した方には参考になるのではないでしょうか。MZ-1500のPSGはいじりまわせばもっともっと面白いことができるはずです。ここはひとつユーザーの奮起に期待したいところです。

体感ゲームミュージック

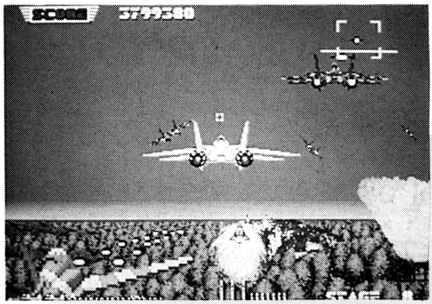
お次はX1用のAFTER BURNERです。このゲームはもうお馴染みですね。セガの体感ゲームの中でもぐるんぐるんと上下左右に振り回されるというとんでもないゲームでした。画面のほうも超高速でビュンビュン背景が流れるわ，ドッグファイトするわでとにかく派手なゲームです。もっとも，最近はギャラクシーフォースという，プレイするのも恥ずかしいようなもの凄いゲームも出ていますからそれほど目立たなくなりましたが。

今回掲載するのはAFTER BURNERというゲームの「AFTER BURNER」という曲です。このゲームのメインテーマとでもいうべき名曲なのですが，もとがサンプリング音源を駆使した音色ですので，はたしてFM音源で再現することができるか，という難曲でもあったわけです。一部MMLの書き換えを行っていますが，プログラム自体はOh!MZ1987年8月号で発表されたMMLでも，単行本『試験に出るX1』に掲載されたものでも共通に使用できるようになっています。また，このプログラムではオクターブ0を使用したところもありますので，X-KEYBOARD上で演奏するときは若干画面が乱れます。

先月に引き続き金子君の作品ですが，彼はまだ受験生，自動車免許もいいけれど，もう浪人はしないように。

SQUAREをMZで

最後はMZ-2500用TRUTHです。SQUAREの曲を6音で演奏するのはキツいかなと思いましたが，なかなか雰囲気を出しているようですね。この曲は皆さんもどこかで聞いたことがあるかもしれません。フジテレビのF1中継のテーマ曲としても有名ですが，正確にはアルバム「TRUTH」のA面5番目の曲ということです。



©SEGA

なまじボーカルつきの曲などよりも，こういったインストルメンタルのほうがFM音源には向いていそうですね。フェードアウトの処理なども自然に決まっています。この曲に限ったことではありませんが，ミュージックプログラムはできるだけオーディオなどに接続してから聞くようにしてください。

なお，このプログラムを演奏するには前もってMMLを拡張し，FM77AVの音色データをセットしておく必要があります（Oh!MZ1987年9月号参照)。もちろん，Super KEYBOARDにも対応します。

リスト1　TECHNOPOLIS/邂逅

```
10 LIMIT$FDFF
20 DATA$FE,$02,$CA,$0A,$FE,$FE,$06,$C2
30 DATA$0B,$FE,$3C,$21,$F2,$37,$C3,$80
40 DATA$3B
50 FORI=0TO16
60 READ D
70 POKE@$FE00+I,D
80 NEXT:POKE$3B7D,$C3,$0,$FE
90 POKE$3C31,$0,$1A
100 REM GOTO880
110 TEMPO6:GOSUB3930:GOSUB1700
120 MUSIC A1$;B1$;C1$;D1$;E1$;F1$
130 MUSIC A2$;B3$;C2$;B2$;E2$;F2$
140 MUSIC A1$;B1$;C2$;D3$;E1$;F2$
150 MUSIC A2$;B4$;C2$;D4$;E2$;F2$
160 MUSIC A5$;B2$;C2$;D5$;E5$;F2$
170 MUSIC A5$;B2$;C2$;D5$;E5$;F2$
180 MUSIC A5$;B2$;C2$;D7$;E5$;F2$
190 MUSIC AT$;BT$;C8$;D8$;E8$;F2$
200 MUSIC A9$;B9$;C9$;D9$;E9$;F2$
210 MUSIC AA$;B9$;C9$;DA$;E9$;F2$
220 MUSIC A9$;B9$;C9$;D9$;E9$;F2$
230 MUSIC AA$;B9$;CC$;DA$;E9$;F2$
240 MUSIC AD$;BD$;C9$;DD$;ED$;FD$
250 MUSIC AE$;BE$;C9$;DE$;EE$;FD$
260 MUSIC AD$;BD$;C9$;DD$;EF$;FD$
270 MUSIC AG$;BG$;C9$;DG$;EG$;FD$
280 MUSIC AH$;BH$;C9$;DH$;EH$;F2$
290 MUSIC AI$;BI$;C9$;DI$;EI$;F2$
300 MUSIC A9$;B9$;C9$;D9$;E9$;F2$
310 MUSIC AA$;B9$;C9$;DA$;E9$;F2$
320 MUSIC A9$;B9$;C9$;D9$;E9$;F2$
330 MUSIC AA$;B9$;CJ$;DA$;E9$;F2$
340 MUSIC AD$;BD$;C9$;DJ$;ED$;FD$
350 MUSIC AE$;BE$;C9$;DK$;EE$;FD$
360 MUSIC AD$;BD$;C9$;DJ$;EF$;FD$
370 MUSIC AG$;BG$;C9$;DL$;EG$;FD$
380 MUSIC AH$;BH$;C9$;DM$;EH$;F2$
390 MUSIC AI$;BI$;C9$;DI$;EI$;F2$
400 MUSIC B2$;B1$;C2$;B2$;E1$;F2$
410 MUSIC B2$;B3$;C2$;B2$;E2$;F2$
420 MUSIC A1$;B1$;C2$;B2$;E1$;F2$
430 MUSIC A2$;B3$;C2$;B2$;E2$;F2$
440 MUSIC A1$;B1$;C2$;D3$;E1$;F2$
450 MUSIC A2$;B4$;C2$;D4$;E2$;F2$
460 MUSIC A5$;B2$;C2$;D5$;E5$;F2$
470 MUSIC A5$;B2$;C2$;D5$;E5$;F2$
480 MUSIC A5$;B2$;C2$;D7$;E5$;F2$
```

```
490 MUSIC AT$;BT$;CK$;D8$;E8$;F2$
500 MUSIC A9$;B9$;C9$;D9$;E9$;F2$
510 MUSIC AA$;B9$;C9$;DA$;E9$;F2$
520 MUSIC A9$;B9$;C9$;D9$;E9$;F2$
530 MUSIC AA$;B9$;CL$;DA$;E9$;F2$
540 MUSIC AJ$;BD$;C9$;DJ$;ED$;FD$
550 MUSIC AE$;BE$;C9$;DK$;EE$;FD$
560 MUSIC AJ$;BD$;C9$;DJ$;EF$;FD$
570 MUSIC AG$;BG$;C9$;DL$;EG$;FD$
580 MUSIC AH$;BH$;C9$;DM$;EH$;F2$
590 MUSIC AI$;BI$;C9$;DI$;EI$;F2$
600 MUSIC A1$;B1$;C2$;D3$;EJ$;F2$
610 MUSIC A2$;B3$;C2$;D4$;EK$;F2$
620 MUSIC A1$;B1$;C2$;D3$;EJ$;F2$
630 MUSIC A2$;B4$;CM$;D4$;EK$;FE$
640 MUSIC AK$;BD$;C9$;DJ$;ED$;FD$
650 MUSIC AL$;BE$;C9$;DK$;EE$;FD$
660 MUSIC AK$;BD$;C9$;DJ$;EF$;FD$
670 MUSIC AM$;BG$;C9$;DL$;EG$;FD$
680 MUSIC A1$;B1$;C2$;D3$;EJ$;F2$
690 MUSIC A2$;B3$;C2$;D4$;EK$;F2$
700 MUSIC A1$;B1$;C2$;D3$;EJ$;F2$
710 MUSIC A2$;B4$;CN$;D4$;EK$;FE$
720 MUSIC AK$;BD$;C9$;DJ$;ED$;FD$
730 MUSIC AL$;BE$;CO$;DK$;EE$;FD$
740 MUSIC AK$;BD$;C9$;DJ$;EF$;FD$
750 MUSIC AM$;BG$;C9$;DL$;EG$;FD$
760 MUSIC A1$;B1$;C2$;D3$;EJ$;F2$
770 MUSIC A2$;B3$;C2$;D4$;EK$;F2$
780 MUSIC A1$;B1$;C2$;D3$;EJ$;F2$
790 MUSIC A2$;B4$;CP$;D4$;EK$;FE$
800 MUSIC AK$;BD$;C9$;DJ$;ED$;FD$
810 MUSIC AL$;BE$;CQ$;DK$;EE$;FD$
820 MUSIC AK$;BD$;C9$;DJ$;EF$;FD$
830 MUSIC AM$;BG$;C9$;DL$;EG$;FD$
840 MUSIC AX$;BX$;C2$;DX$;EX$;F2$
850 MUSIC AY$;BY$;C2$;DY$;EY$;F2$
860 MUSIC AZ$;BZ$;CZ$;DZ$;EZ$;FZ$
870 MUSICWAIT:GOSUB4010
880 POKE$3C31,$0,$1D:GOSUB2670
890 MUSIC A1$;A1$;A1$;A1$;A1$;A1$
900 MUSIC D2$;B2$;C1$;A2$;E2$;F1$
910 MUSIC D1$;B1$;C1$;A1$;E1$;F1$
920 MUSIC D1$;B1$;C3$;A1$;E3$;F3$
930 MUSIC D1$;B1$;C3$;A1$;E4$;F3$
940 MUSIC D1$;B5$;C3$;A1$;E3$;F3$
950 MUSIC D6$;B6$;C6$;A3$;E4$;F6$
960 MUSIC D7$;B5$;C7$;A1$;E3$;F7$
970 MUSIC D8$;B6$;C7$;A1$;E4$;F7$
980 MUSIC D7$;B5$;C7$;A1$;E3$;F7$
990 MUSIC D8$;B6$;C8$;A3$;E4$;F8$
1000 MUSIC D7$;B5$;C7$;A4$;E3$;F7$
1010 MUSIC D8$;B6$;C7$;A5$;E4$;F7$
1020 MUSIC D7$;B5$;C7$;A4$;E3$;F7$
1030 MUSIC D9$;B6$;C9$;A6$;E9$;F9$
1040 MUSIC DA$;BA$;CA$;AA$;EA$;FA$
1050 MUSIC DB$;BB$;CA$;AB$;EB$;FB$
1060 MUSIC DA$;BC$;CA$;AC$;EA$;FA$
1070 MUSIC DB$;BD$;CA$;AB$;EB$;FB$
1080 MUSIC DC$;BE$;CA$;AD$;EC$;FA$
1090 MUSIC DB$;BF$;CA$;AE$;EB$;FB$
1100 MUSIC DC$;BG$;CA$;AF$;EC$;FA$
1110 MUSIC DD$;BH$;CC$;AG$;ED$;FC$
1120 MUSIC D1$;B5$;CA$;A1$;E3$;FA$
1130 MUSIC D6$;B6$;CD$;A3$;E4$;FD$
1140 MUSIC D7$;B5$;CA$;A1$;E3$;FA$
1150 MUSIC D8$;B6$;CA$;A1$;E4$;FA$
1160 MUSIC D7$;B5$;CA$;A1$;E3$;FA$
1170 MUSIC D8$;B6$;CA$;A3$;E4$;FA$
1180 MUSIC D7$;B5$;CA$;A4$;E3$;FA$
1190 MUSIC D8$;B6$;CA$;A5$;E4$;FA$
1200 MUSIC D7$;B5$;CA$;A4$;E3$;FA$
1210 MUSIC D9$;B6$;C9$;A6$;E9$;F9$
1220 MUSIC DA$;BI$;CA$;AA$;EA$;FA$
1230 MUSIC DB$;BB$;CA$;AB$;EB$;FB$
1240 MUSIC DA$;BJ$;CA$;AC$;EA$;FA$
1250 MUSIC DB$;BK$;CA$;AB$;EB$;FB$
1260 MUSIC DC$;BL$;CA$;AD$;EC$;FA$
1270 MUSIC DB$;BF$;CA$;AE$;EB$;FB$
1280 MUSIC DC$;BG$;CA$;AF$;EC$;FA$
1290 MUSIC DF$;BH$;CE$;AG$;EE$;FE$
1300 MUSIC QD$;QB$;QC$;QA$;QE$;QF$
1310 MUSIC RD$;RB$;RC$;RA$;RE$;RF$
1320 MUSIC DB$;BB$;CA$;AB$;EB$;FG$
1330 MUSIC DA$;BC$;CA$;AC$;EA$;FA$
1340 MUSIC DB$;BD$;CA$;AB$;EB$;FG$
1350 MUSIC DC$;BE$;CA$;AD$;EC$;FA$
1360 MUSIC DB$;BF$;CA$;AE$;EB$;FG$
1370 MUSIC DC$;BG$;CA$;AF$;EC$;FA$
1380 MUSIC DG$;BH$;CF$;AG$;EF$;FF$
1390 MUSIC D7$;B5$;C7$;A4$;E3$;F7$
1400 MUSIC D8$;B6$;C7$;A5$;E4$;F7$
1410 MUSIC D7$;B5$;C7$;A4$;E3$;F7$
1420 MUSIC D8$;B6$;CL$;A5$;E4$;FL$
1430 MUSIC D7$;B5$;C7$;A4$;E3$;F7$
1440 MUSIC D8$;B6$;C7$;A5$;E4$;F7$
1450 MUSIC D7$;B5$;C7$;A4$;E3$;F7$
1460 MUSIC DH$;B6$;CL$;A5$;E4$;FL$
1470 MUSIC D7$;B5$;C7$;A4$;E3$;F7$
1480 MUSIC D8$;B6$;C7$;A5$;E4$;F7$
1490 MUSIC D7$;B5$;C7$;A4$;E3$;F7$
1500 MUSIC DI$;B6$;CL$;A5$;E4$;FL$
1510 MUSIC DJ$;B5$;C7$;A4$;E3$;F7$
1520 MUSIC DK$;B6$;C7$;A5$;E4$;F7$
1530 MUSIC DL$;BN$;C7$;A4$;E3$;F7$
1540 MUSIC DM$;BO$;CM$;A5$;E4$;FM$
1550 MUSIC DL$;BN$;CH$;A4$;EG$;FH$
1560 MUSIC DM$;BO$;CI$;A5$;EH$;FI$
1570 MUSIC DL$;BN$;CJ$;A4$;EI$;FJ$
1580 MUSIC DM$;BO$;CK$;A5$;EJ$;FK$
1590 MUSIC DL$;BN$;"" ;AW$;EW$
1600 POKE$3B7D,$21,$F2,$37
1610 MUSIC DM$;BO$;CO$;AX$;EX$;CO$
1620 MUSIC DL$+DM$+DN$;BN$+BO$+BN$;CP$+CQ$+CP$;AY$+AZ$+AY$;EY$+E
Z$+EY$;CP$+CQ$+CR$
1630 FOR I=$1E TO $2A
1640 FOR O=1 TO $FF STEP 3
1650 POKE$3C31,O,I
1660 NEXT O
1670 NEXT I
1680 POKE$3C31,$A5,$22
1690 CCOLOR ,,7:PRINT "END":END
1700 REM
1710 A1$="V1202S0M5#A1#A#A#A#A#A#A#ABBBBBBB#A#A#A#A#A#A#ABBBB
BBBB
1720 C1$="V1006S0M1B3B1BB3B1BB3B1BB3B1BB3B1BB3B1BB3B1BB3B1B
1730 D1$="V1204S0M20R9R
1740 E1$="V1101S0M12G3+GRGG+GRGG+GRGG+GRG
1750 F1$="V1503S0M1G1R4G1R4G1R4G1R4G1R4G1R4G1R4G1R4
1760 A2$="FFFFFFF#F#F#F#F#F#F#FFFFFFFF#F#F#F#F#F#F#F
1770 B2$="R9R
1780 C2$="M1V10B3B1BB3B1BB3B1BB3B1BB3B1BB3B1BB3B1BB3B1B
1790 E2$="D3+DRDD+DRDD+DRDD+DRD
1800 F2$="G1R4G1R4G1R4G1R4G1R4G1R4G1R4G1R4
1810 A3$="V1202S6M1#A7B#AB
1820 D3$="V1204S0M26F6D3G6D3F6D3G6D3
1830 D4$="C6-A3D6-A3C6-A3D7"
1840 A5$="O3S0M7G3G1GG3G1GG3G1GG3G1GG3G1GG3G1GG3G1GG3G1G
1850 D5$="S0M9O3G2R0F3G2R0#A2R0R7#D3R6F3R6
1860 E5$="G3+GDFR+FD+DC+CC+CR+DD+F"
1870 D7$="G2R0F3G2R0#A2R0R7#D3RRF3R7
1880 A8$="V1002S0M12#A3#G#A+#CR+CR#GR#A#A8
1890 AT$="V1402S0M20 G5G5R3A5RG8R3
1900 B8$="V10M12O3D3CDFRERCRDD8
1910 BT$="V1402S0M20 G3F5#A7F6G8
1920 C8$="V10B3B1BB3B1BB3B1BB3B1BB3B1BB3B1RM2V15BBBRBBBR
1930 D8$="O2S0M12D3CDFRERCRDD8
1940 E8$="O1G3+GG#DRDRDRCC8
1950 A9$="V13S0M6O2R5G3A#A+C+D+#D+D1+#D+D8R3
1960 B9$="V10S0M9O2D9D
1970 C9$="M1V15O3G5O6M2BO3M1GO6M2BO3M1GO6M2BO3M1GO6M2B
1980 D9$="V10S0M9O2G9F
1990 E9$="-A1RG3-A1RG3RG-A1RG3-A1RG3-A1RG3RG-A1RG3
2000 AA$="R5+C7D3V12DV13#DV12#DV13+C#A5+C
2010 DA$="#D9F
2020 CC$="V15O3G5O6BO3GO6BO3GO6BB1BBBBBBR
2030 AD$="O3M8#A3AGF5DFG3G7A5
2040 BD$="O2M6V10G6FR3D5C3C7D5
2050 DD$="#A6AR3F5#D3#D7F5
2060 ED$="V11M20G6F7D5C3C5C3CD3D
2070 FD$="G3GGGGRGGGGGGRGG
2080 AE$="#A3AGF5D3F+D8R3+C5
2090 BE$="G6FF5#D8F5
2100 DE$="#A6AA5G8A5
2110 EE$="G6F6F5#D7#D5F
2120 EF$=ED$
2130 AG$="#A3AGF5D3F+D7R3+#F7
2140 BG$="G6FF5#D7D
2150 DG$="#A6AA5G7#F
2160 EG$="G6F6F5#D1R#D3#D1R#D3D5#F
2170 AH$="V13S0M2002R5G3ABO3CD7-B3CD#DFG
2180 AI$="#A6AF3DG6#F7R3
2190 BH$="V13S0M2002G7G3ABO3CD7-B3CD#D
2200 BI$="#D6-#A7R3#D6D7R3
2210 DH$="V13S0M2002B8G3AB+C+D7B3+C
2220 DI$="G6-D7R3G6#F7R3
2230 EH$="M7G3+GG+GR+GG+GF+FF+FR+FF+F
2240 EI$="#D3+#DR-#A-#A-#AF-B#D#DDDRD-#A-A
2250 CJ$="V15O3G5O6BO3GO6BO3G3O6B1BBBBRBBBBRRBB
2260 DJ$="V12O2G1FGRGFG#AGFGRAFDFGFGRGFG#AGFG#AAFDF
2270 DK$="G1FGRGFG#AGFGRAFDFGFGR+D+#A+D+C#A+C#AAGAGF
2280 DL$="G1FGRGFG#AGFGRAFDFGFGR+D+#A+D+C#A+CA#A+C+D+E+#F
2290 DM$="+G1+F+GRV13S0M2002R5G3AB+C+D7B3+C
2300 B1$="V15S0M6O1R9R3#AAFRFGD
2310 B3$="R9R3FDCRD0RR4D0RRR
2320 B4$="R9R3FDCRD0RR4#F0RRR
2330 CK$="V15O3G5O6BO3GO6BO3GO6BB3BBB
2340 CL$="V15O3G5O6BO3GO6BO3GO6BB1BBBRRBB
2350 AJ$="O3M17#A3AGF5DFG3G7A5
2360 CM$="V10M1B3B1BB3B1BB3B1BB3B1BV15O3G1O6M2RBBRRBBO3M1GO6M2BB
BBBBR
2370 CN$="V10M1B3B1BB3B1BB3B1BB3B1BB3B1BB3B1BV15M2BBBBBO3M1GO6M2
BB
2380 AK$="O3M30#A3A2R0G2R0F5DFG1RG7A5
2390 AL$="#A3A2R0G2R0F5D3FS8+D7S0+D6+C5
2400 AM$="#A3A2R0G2R0F5D3FS8+D5S0+D6+#F7
2410 CO$="M1V15O3G5O6M2BO3M1GO6M2BO3M1GO6M2BO3M1GO6M2B1BBR
2420 CP$="V10M1B3B1BB3B1BB3B1BB3B1BB3B1BB3B1BV15O3GRO6M2BBO3M1GR
O6M2BB
2430 CQ$="M1V15O3G5O6M2BO3M1GO6M2BO3M1GO6M2BB1BBRBBBR
2440 AX$="V1202S0M20#A7V11BV10#AV9B
2450 AY$="V8FV7#FV6FV5#F
2460 AZ$="V4#AV3BV2#AV1B
2470 BX$="V13S0M6O1R9R3#AAFRV12FGD
2480 BY$="R9R3V7FDCRD0RR4D0RRR
2490 BZ$="R9R3V3#AAFRV2FGD
2500 CZ$="M1V9B3B1BB3V7B1BB3B1BB3V5B1BB3B1BB3V3B1BB3B1BB3V1B1B
2510 DX$="V1204S0M26F6D3V11G6D3V10F6D3V9G6D3
2520 DY$="V8C6-A3V7D6-A3V6C6-A3V5D7
2530 DZ$="V4F6D3V3G6D3V2F6D3V1G6D3
2540 EX$="V1301S0M7G1RG3G1RG3V12RG3G1RG3V11G1RG3G1RG3V10RG3G1RG3
2550 EY$="V9D1RD3D1RD3V8RD3D1RD3V7D1RD3D1RD3V6RD3D1RD3
```

```
2560 EZ$="V5G1RG3G1RG3V3RG3G1RG3V2G1RG3G1RG3V1RG3G1RG3D1RD3
2570 FZ$="V15G1R4V13G1R4V11G1R4V9G1R4V7G1R4V5G1R4V3G1R4V1G1R4G1
2580 FE$="G1R4G1R4G1R4G1S5M1V12AA05CV13EGBV1404GGB05DV15F06CFB03
S0M1
2590 EJ$="V1301S0M7G3GG1RG3RGG1RG3GGG1RG3RGGG
2600 EK$="D3DD1RD3RDD1RD3DDD1RD3RDDD
2610 TI$="000000"
2620 SOUND=(7,0):SOUND=(6,3)
2630 SOUND=(15,5):SOUND=(14,7)
2640 FORI=0TO300
2650 SOUND=(4,I):SOUND=(12,I)
2660 NEXT:RETURN
2670 A1$="V13S0M2503
2680 B1$="V14S0M1004
2690 C1$="V15S0M103
2700 D1$="V15S8M3002
2710 E1$="V15S0M201
2720 F1$="V15S0M103
2730 A2$="O0L1A+A02A+A04A+A06ARRRRRRRR
2740 B2$="O0L1RRRRRA+A02A+A04A+A06ARRR
2750 D2$="O0L1RRRRA+A02A+A04A+A06ARRRRR
2760 E2$="O0L1RRRRRRRRA+A02A+A04A+A06A
2770 E3$=E1$+"F1CFFR5G1DGGR5D1-ADDR5E1-BEER5
2780 C3$="03M1L1GRRGRRRRGRRGRRRRGRRGRRRRGR06M1B0BBRM4B503M1
2790 F3$=C3$
2800 E4$="F1CFFR5G1DGGR5E1-BEER5A1EAA
2810 B5$=B1$+"A7DFG"
2820 B6$="A7DG+C"
2830 C6$="03M1L1GRRGRRRRGRRGRRRR06M2BBB03M1GRRRR06M3BBBBBBBR
2840 F6$="L1GRRGRRRRGRRGRRRRGRRG06M2BBBRM3BBBBBBBR
2850 D6$="R9R8D1EDC"
2860 D7$="-A1RCDD8D7D4R1DEDC
2870 D8$="-A1RC-A-A8-A7-A4R1DEDC
2880 C7$="03M1G1RRG06M3B503M1G1RRG06M3B503M1G1RRG06M3B503M1G1RRG
06M3B5
2890 F7$="03M1G1RRG06M2B503M1G1RRG06M2B503M1G1RRG06M2B303M1GG1RR
G06M2B5
2900 C8$="03M1G1RRG06M3B503M1G1RRG06M3B503M1G1RRG06M3B503M1G1RRG
06M2BV12BV15BV12BV15
2910 F8$="03M2G1RRG06B503G1RRG06B503G1RRG06B303GG1RRG06V12BV15BV
12BV15B
2920 A3$="R9"+E2$
2930 A4$=A1$+"R5-ACAG6E3D7
2940 A5$="R5-ACEDE3CC7
2950 A6$="R5-ACEDC3-A-A5-B
2960 C9$="03M1G1RRG06M3B503M1G1RRG06M3B503M1G1RRG06M2G1GGRM3BRBR
BBBR
2970 F9$="03M1G1RRG06M2B503M1G1RRG06M2B503M1G1RRG06M1G1GGRM3BRBR
BBBR
2980 D9$="-A1RC-A-A8-A8V12-G5
2990 E9$="F1CFFR5G1DGGR5E1-BEER5-A1R-AR-B-B-BR
3000 AA$="C8R5R9
3010 DA$="02V11S8C9C
3020 BA$="S0M20V1202E8R5 V14G5E3D6C5
3030 EA$="S0M102V12E1CEGGCEGEECGCGCEGGECECCGGEECCEGE
3040 CA$="03M1G1RRG06M3B503M1G1R406M3B503M1G1RRG06M3B503M1G1RGRO
6M3B5
3050 FA$="03M1G1RRG06M2B303M1G1RGR406M2B303M1G1RGRRG06M2B303M1G1
RGRGR06M2B5
3060 EB$="A1FA+C+CFA+CAAF+CF+CFA +C+CAFAFF+C+CAAFFA+CA
3070 BB$="S8-A7S0M5-A3EEM20ED9
3080 AB$="03S0R9R6A5G3E5
3090 DB$="02F9F
3100 FB$="03M1G1RRG06M2B303M1G1RGRRR06M2B303M1G1RG1RRG06M2B303M1
G1RGRGRM4B3B
3110 CC$="03M1G1RRG06M3B504M2E1EERV12DDDRV15EEEEEEEEDRRR06M3BBBB
3120 FC$="03M1G1RRG06M2B303M1G1R04M2V12E1EERV15DDDREEEEEEEEDRRRO
6BBBB
3130 DC$="E9E
3140 DD$="F8G8R7
3150 EC$="G1EGBBEGBGGEBEBEG BBGEGEEBBGGEEGBG
3160 ED$="A1FA+C+CFA+CAAF+CG+DGB +D+DBGBGG+D+DRRRRRRR
3170 ED$="A1FA+C04S8E3-A5C3G802E3#DD#C
3180 AC$="C9R
3190 AD$="D9R
3200 AE$="R9R6E5D3C5
3210 AF$="-G9R
3220 BC$="R9G3RE2R0D6C5
3230 BD$="-A8ER5E
3240 BE$="D5E3S8G6S0G9E5
3250 BF$="D6-AE8R5E5
3260 BG$="D5E3A5A3G8R5E
3270 BH$="D5E3-A5C3D8R7
3280 AG$="S8R505E3-A5C3G8R7
3290 CD$="03M1L1GRRG06M3B503M1G1RRG06M3B503M1G1RRG06M3B04M2EERV1
2DDDRV1506M3BBBB
3300 FD$="03M1L1GRRG06M2B503M1G1RRG06M2B503M1G1RRG06M2B5V15D1DDR
V1506BBBB
3310 BI$="S0M20V1202E8R5 V14G4R1E3D6C5
3320 BJ$="R9G5E3D6C5
3330 BK$="-A8-A3E8R3E5
3340 BL$="D5E3S8G6S0G9E2R0E3
3350 DF$="F8S8#A9R5
3360 EE$="A1FA+C04S8E3-A5C3G8R7
3370 CE$="03M1G1RRG06M3B503M1G1RRR06M3B503M1G1RRG04M2EEERV12DDDR
06M3V15BBRB
3380 FE$="03M1G1RRG06M2B303M1G1RGRRR06M2B303M1G1RGRRGV1206M3B04M
2EERV15DDDR06M3BBRB
3390 AH$="04S8B1#AV9A#GV7G#FV5FEV3#DDV1#CRR5R9V13
3400 BM$="S0M20V1103+C1BV9#AAV7#GGV5#FFV3E#DV1D#CR5V1402G5E3D6C5
3410 QA$=LEFT$(AH$,34):RA$=MID$(AH$,35)
3420 QB$=LEFT$(BM$,41):RB$=MID$(BM$,42)
3430 QC$=LEFT$(CA$,29):RC$=MID$(CA$,30)
3440 QE$=LEFT$(EA$,26):RE$=MID$(EA$,27)
3450 QF$=LEFT$(FA$,38):RF$=MID$(FA$,39)
3460 QD$="02V11S8+C1BV9#AAV7#GGV5#FFV3E#DV1D#CR5V11
3470 RD$="02C9
3480 EF$="A1FA+C04S8E3-A5C3G8R7
3490 CF$="03M1G1RRG06M3B303M1G1RGR404M2D1DV12DV15REEEEEEEEERRR06
M3BBBB
3500 FF$="03M1G1RRG06M2B303M1G1RGR404M2V12D1DV15DREEEEEEEEERRR06
M2BBBB
3510 FG$=LEFT$(FB$,LEN(FB$)-5)+"V1306M5A1AAAV15
3520 DG$="F8G8R5V15S802D1EDC
3530 DH$="-A1RC-A-A7-A3 G1#GA4#G1A4#G1A5 03D1EDC
3540 DI$="02A1R+CAV14A7V13A3G1#GV12A4#G1V11A4#G1A5V1003D1EDC
3550 DJ$="-A1RCDV9D8V8D7V7D4R1V6DEDC
3560 DK$="-A1RV5C-AV4-A6V3-A6V2-A7V1-A4
3570 BN$="V14S0M2004A9F
3580 DL$="V8S0M2004+C7V14D9G7
3590 BO$="A9G
3600 DM$="V8G7V14D9+C7
3610 EG$="V14S0M201F1CFFR5G1DGGR5V13D1-ADDR5E1-BEER5
3620 EH$="V12F1CFFR5G1DGGR5V11E1-BEER5A1EAAR5
3630 EI$="V10F1CFFR5G1DGGR5V9D1-ADDR5E1-BEER5
3640 CH$="03M1G1RRG06M3B5V1403M1G1RRG06M3B5V1303M1G1RRG06M3B5V12
03M1G1RRG06M3B5
3650 CI$="V1103M1G1RRG06M3B503M1G1RRG06M3B5V1003M1G1RRG06M3B503M
1G1RRG06M3B5
3660 CJ$="03M1G1RRG06M3B5V903M1G1RRG06M3B5V803M1G1RRG06M3B503M1G
1RRG06M3B5
3670 CK$="03M1G1RRG06M3B503M1G1RRG06M3B504M2E1EERDDDRDDDR06M3BBB
B
3680 FH$="03M1G1RRG06M2B5V1403M1G1RRG06M2B5V1303M1G1RRG06M2B3V12
03M1GG1RRG06M2B5
3690 FI$="V1103M1G1RRG06M2B503M1G1RRG06M2B5V1003M1G1RRG06M2B303M
1GG1RRG06M2B5
3700 FJ$="03M1G1RRG06M2B5V903M1G1RRG06M2B5V803M1G1RRG06M2B303M1G
G1RRG06M2B5
3710 FK$="03M1G1RRG06M2B503M1G1RRG06M2B504M2E1EERDDDRDDDR06M2BBB
B
3720 CL$="03M1G1RRG06M3B503M1G1RRG06M3B503M1G1RRG06M3B504M2E1EEV
12E06M3V15BBBB
3730 FL$="03M1G1RRG06M2B503M1G1RRG06M2B503M1G1RRG06M2B303M1G1R04
M2EEV12EV15E06M2BBBB
3740 CM$="03M1G1RRG06M3B503M1G1RRG06M3B504M2E1EERDDDRR506M3B1BBB
3750 FM$="03M1G1RRG06M2B503M1G1RRG06M2B504M2R5D1DDRDDDR06M2BBBB
3760 EJ$="V8F1CFFR5V7G1DGGR5V6E1-BEER5V4A1EAAR5
3770 BP$="V12A9V10F
3780 CP$="E7-GCD
3790 CO$="04S0M7E7-GCE
3800 BQ$="V8A9V6G
3810 CQ$=CO$
3820 CR$=CP$+"9
3830 DN$=DL$+"V8G5
3840 AW$="V13S0M1203R5-A7A7R3E7R3
3850 AX$="R5-A7EE5A7
3860 AY$="R5-A7A7R3E7R3
3870 AZ$="R5-A7EE5A7
3880 EW$="V13S0M1203R5-AC7GD7
3890 EX$="R5-AC7D6C7R3
3900 EY$="R5-AC7GD7
3910 EZ$="R5-AC7D6C7R3
3920 TI$="000000":RETURN
3930 INIT"CRT:G":CLS3:CCOLOR,,4
3940 CURSOR 8,10
3950 PRINT[2]"T E C H N O P O L I S"
3960 CURSOR 7,13
3970 PRINT"Y E L L O W   M A G I C"
3980 CURSOR 10,16
3990 PRINT"O R C H E S T R A"
4000 RETURN
4010 CURSOR 8,10
4020 PRINT SPC(21)
4030 CURSOR 12,10
4040 PRINT[5]"K A I - K O H"
4050 CCOLOR,,6
4060 GOTO3960
```

リスト2　AFTER BURNER音色定義

```
10 '   After Burner      by  SEGA
20 '     Sound data
30 '                     by S.Kaneko
40 '
50 '                     in 27th Feb. 88'
60 '
70 CLR :RESTORE 190 :GOSUB "Set tone"
80 RUN "After Burner"
90 '
100 '     I1=コンコン     I2=Bass      I3=D-Guiter I4=B-Drum
110 '     I5=Snare       I6=チャラ チャラ  I7=Strings  I8=Hi-Hat
120 '     I9=I-Guiter
```

▶確か，３年ほど前だろうか。ＰＳＧのエンベロープをいじくって出た音に感動していたのは……。マドンナ，安全地帯など多くの曲をスコアとにらめっこして演奏していた。そして３年後の現在，なんだ，この進歩の速さは！　凄い，なんていい音を出すんだ。

大石　信雄（18）東京都

```
130 '
140    LABEL "Set tone"
150  FOR I=0TO20 :READ A$ :FOR J=0TO15 :X=VAL("&h"+MID$(A$,1+3*J,2))
160  POKE &HB190+J+I*16,X :Z=Z+X :NEXT J :NEXT I
170  IF Z=18190 THEN RETURN ELSE PRINT"DATA ERROR" :STOP
180 '
190 DATA FC 00 36 31 76 71 1D 08 20 06 1C 1C 1F 1B 13 91
200 DATA 12 11 00 00 05 05 F4 F7 F4 F7 00 00 00 80 00 00
210 DATA 80 00 00 00 E0 00 46 45 40 41 1D 2F 1D 00 DF DF
220 DATA 9A 9F 07 06 09 88 07 06 06 04 29 19 19 39 80 00
230 DATA 00 00 00 00 80 00 00 00 D5 20 41 41 41 41 00 00
240 DATA 00 00 1F 9F 5F 9F 82 81 82 82 05 01 01 01 07 35
250 DATA 37 35 00 03 80 81 00 C8 B2 00 02 00 F0 00 01 0F
260 DATA 01 40 00 2B 07 00 1F 1E 1F 1F 1A 1C 0D 8E 40 C0
270 DATA 05 0A FD FE C8 37 00 00 00 00 D0 00 80 00 00 00
280 DATA FC 00 0C 01 00 01 00 00 13 00 1F 5C 5E 9F 00 91
290 DATA 11 8F C0 00 80 00 01 F8 D9 F8 00 00 00 00 F4 00
300 DATA 80 00 00 00 D3 00 03 04 02 41 25 21 1B 00 1E 52
310 DATA 59 9C 14 14 0A 0E 00 00 00 00 F8 75 F3 F7 04 06
320 DATA 00 00 00 C8 80 00 02 00 EC 33 32 30 72 50 11 00
330 DATA 10 00 5B 14 5A 14 08 02 05 00 01 01 01 01 F4 05
340 DATA F5 07 00 00 00 00 00 C8 9E 00 02 01 FB 00 0E 06
350 DATA 07 00 0F 1B 11 05 1A 1A 1A 16 04 08 16 92 40 40
360 DATA 80 00 32 72 BA F8 00 00 00 00 00 C8 80 00 02 00
370 DATA E4 00 30 32 70 72 00 00 00 00 14 19 14 17 02 03
380 DATA 01 04 05 01 01 00 F6 F7 F6 F7 04 01 02 01 00 00
390 DATA 80 00 02 00 00 00 00 00 00 00 00 00 00 00 00 00
```

リスト3 AFTER BURNER

```
10 '   After Burner      by  SEGA
20 '     Music data
30 '                     by S.Kaneko
40 '
50 '                     in 27th Feb. 88'
60 '
70    CLS 0 :TEMPO 0 :TIME=0
80    VWORK=&HAE69 :TCMD=&HAC40 :OFST=0
90    IF PEEK(VWORK) THEN TCMD=&HAC3F ELSE VWORK=&HAE7C :OFST=3
100   POKE TCMD,&H18 :POKE TCMD+1,&H1B
110   CTC=PEEK(&HA9D7+OFST)*256+PEEK(&HA9D6+OFST)
120   OUT CTC,7 :OUT CTC,4
130   '
140   R5$="Y33,188" :C5$="Y33,252" :L5$="Y33,124"
150   R6$="Y36,147" :C6$="Y36,211" :L6$="Y36,83"
160   D1$="O4 A>GA<A>EG<A>DE<A>CD<A>C<BG"
170   E$=R6$+"CEGA"+L6$+"CEGB"+R6$+"EGA>C"+L6$+"<EAB>D"+R6$+"<EGB
>C"
180   E1$=E$+L6$+"<GB>DF"+R6$+"<A>CEG"+L6$+"<A>CEA"+C6$
190 '
200   "P-In":"P-In4":"P-In3":"P-In5"
210   "P-A" :"P-A2" :"P-A'" :"P-A3" :"P-A5"
220   "P-A'":"P-A2" :"P-A'" :"P-A3" :"P-A5"
230   "P-B" :"P-In2":"P-In4":"P-In3"
240   "P-B2":"P-B3" :"P-B4" :"P-B3'"
250   "P-B5":"P-A4" :"P-B7" :"P-B6" :"P-A4"
260   "P-En"
270 '
280 IF TIME>315 THEN 290 ELSE PAUSE 10 :GOTO 280
290  FOR I=1 TO 25
300   POKE VWORK+1 ,(PEEK(VWORK+1 )-4)
310   POKE VWORK+9 ,(PEEK(VWORK+9 )-3)
320   POKE VWORK+13,(PEEK(VWORK+13)-3)
330  PAUSE 1 :NEXT
340 END
350 '
360      LABEL "P-In"    ' RESCUE LUCY FROM MATCH'S HANDS
370 A$="T82 V121 O0I4L8 R@8E-4"
380 B$=":V127 O3I5L8 R@8D4"
390 C$=":V117 I2L8"
400 D$=":V115 I1L8"
410 E$=":I7L8"
420 F$=":I7L8"
430 G$=":I7L8"
440 H$=":I8L8"
450  "!" '0
460 A$="O0 E-4E-4E-4E-4 E-4E-4E-4E-4"
470 B$=":O3 R@1792 DD"
480 C$=":O3 AAAA AAAA AAA A@43A@42A@43 AAAA"
490 D$=":"+D1$
500  "!" '1,2
510 B$=":O4 R1 D+16D+16<D>R D16D16<DR DD"
520  "!" '3,4
530 B$=":O3 R@1792 DD"
540  "!" '1,2
550 B$=":T81 O4 R@1152 L16 DDR8DD RB32B32BB GGD8 L8"
560 H$=":O4 R V110E V113E V115E V119E V123E V127E V126E V125 EEE
B EEEB"
570  "!" '5,6              ' MATCH IS APPROCHING
580      LABEL "P-In2"    ' EQUIP ENOUGH MISSILE !
590 A$="O0 E-E-R4 E-E-R4 E-E-R4 E-.E-.E-"
600 B$=":O3 R4D4R4D4 R4D4"+R5$+"D."+L5$+"D."+R5$+"D"+C5$
610 C$=":V117 O3 AAAA AAAA AAAA AAGA"
620 D$=":V112 I1"+D1$
630 E$=":V106 O5I7q5A4<I9q8A4RA4GA@496 R@16>I7q5G.G.A"
640 F$=":V106 O5I7q5E4<I9q8E4RE4DE@496 R@16>I7q5D.D.E"
650 G$=":V107 O4I7q5 A4R@1280 G.G.A"
660 H$=":V124 O4 EEEB EEEB EE EE E8.E8.E8"
670  "!" '7,8
680      LABEL "P-In3"
690 C$=":O3 AAAA AAAA AAAA AB>C<G"
700 E$=":O4 R4I9q8A4RA4GA@496 R@16>I7q5A.A.G"
710 F$=":O4 R4I9q8E4RE4DE@496 R@16>I7q5E.E.D"
720 G$=":O4 R1. A.A.G"
730 H$=":O4 REEB EEEB EE EE E8.E8.E8"
740  "!" '9,10
750 C$=":O3 GGGG GGGG GGGG GGGG"
760 D$=":O4 G>G-G<G>EG-<G>DE<G>D-D<GBAG-"
770 E$=":O4 R4I9q8G4RG4FG@496 R@16>I7q5F.F.G"
780 F$=":O4 R4I9q8D4RD4CD@496 R@16>I7q5C.C.D"
790 G$=":O4 R1. F.F.G"
800  "!" '11,12
810  RETURN
820      LABEL "P-In4"
830 C$=":O3 GGGG GGGG GGGG GGAB"
840 E$=":O4 R4I9q8G4RG4FG@496 R@16>I7q5G.G.A"
850 F$=":O4 R4I9q8D4RD4CD@496 R@16>I7q5D.D.E"
860 G$=":O4 R1. G.G.A"
870  "!" '13,14
880 A$="O0 E-E-R4 E-E-R4 E-E-R4 E-.E-.E-"
890 B$=":O3 R4D4R4D4 R4D4"+R5$+"D."+L5$+"D."+R5$+"D"+C5$
900 C$=":O3 AAAA AAAA AAAA AAGA"
910 D$=":"+D1$
920 E$=":O4 R4I9q8A4RA4GA@496 R@16>I7q5G.G.A"
930 F$=":O4 R4I9q8E4RE4DE@496 R@16>I7q5D.D.E"
940 G$=":O4 R1. G.G.A"
950  "!" '7',8
960  RETURN
970      LABEL "P-In5"
980 A$="O0 E-E-R4 E-E-R4 E-E-R4 E-.E-."
990 B$=":O3 R4D4R4D4 R4D4D.D."
1000 C$=":O3 GGGG GGGG GGGG GG"
1010 D$=":O4 G>G-G<G>EG-<G>DE<G>D-D<GB"
1020 E$=":O4 R4I9q8G4RG4FG@496 R@16>I7q5G.G.R"
1030 F$=":O4 R4I9q8D4RD4CD@496 R@16>I7q5D.D.R"
1040 G$=":O4 R1. G.G.R"
1050  "!" '13,15
1060 A$="O0 E-4RE-4E-E-"
1070 B$=":O3 R2D4R4.DR4.D"
1080 C$=":O3 EEEE EEEE EEEE EEEE"
1090 D$=":I7"
1100 E$=":O5 ER@640EE"
1110 F$=":O4 A-R@640A-A-"
1120 G$=":O4 ER@640EE"
1130 H$=":O4 L8 E4EE E4EE E4EE E4EE"
1140  "!" '16,17
1150 A$="O0 E-4RE-4RE-E-"
1160 B$=":O3 RDR2.RDRDRDDD
1170 C$=":O3 EEEE EEEE EEEE EG4."
1180 E$=":O5 RER4ER@640EER"
1190 F$=":O4 RA-R4A-R@640A-A-R"
1200 G$=":O4 RER4ER@640EER"
1210  "!" '18,19
1220  RETURN
1230      LABEL "P-A"     ' MATCH RUNWAY TO THE NORTH!
1240 PLAY "::::I3:I3:I3"
1250     LABEL  "P-A'"
1260 A$="O0 E-E-R4E-4R4E-4R4E-E-R4"
1270 B$=":O3 R4D4R4D4R4D4R4D4"
1280 C$=":V111 O3 AAAA AAAA AAAA AB->C<B-&+"
1290 D$=":V110 O6 D@32E@1760<A>C"
1300 E$=":V94 O3Q7 AAR@640A R4.AR"
1310 F$=":V92 O3Q7 EER@640E R4.ER"
1320 G$=":V94 O2Q7 AAAA AAAA AAAA AB->C<B-"
1330 H$=":O4 EEEB EEEB EEEB EEEB"
1340  "!" '20,21
1350 C$=":O3 B-B-B-B- B-B-B-B- B-B-B-B- B-AFG&+"
1360 D$=":O6 D@14E@370D@1280"
1370 E$=":O3 B-B-R@640B-R4.B-R"
1380 F$=":O3 FFR@640FR4.FR"
1390 G$=":O2 B-B-B-B- B-B-B-B- B-B-B-B- B-B-B-B-"
1400  "!" '22,23
1410 C$=":O3 GGGG GGGG GGGG GAB-F"
1420 D$=":O6 D@26E@358D4.F@896F@24G@104FE"
1430 E$=":O3 GGR@640G R4.GR"
1440 F$=":O3 DDR@640D R4.DR"
```

▶えーい，なんなんだ？　２浪には人権はないのか？　２次募集で滑りこんだ奴とか，あきらめて専門学校にいった奴とか友だちにいるけど，俺は大学に行きたいんでぃ！　来年こそは……って昨年も同じことをいいながらこれ書いてたような気もするけど。大学入ったら免許取って，ビデオ買って……。　田中　正志（19）千葉県

```
1450 G$=":O2 GGGG GGGG GGGG GGGG"
1460  "!" '24,25
1470  RETURN
1480    LABEL "P-A2"
1490 C$=":O3 FFFF FFFE EEFG >DC<BG
1500 D$=":O6 E2RDC<B@1152"
1510 E$=":O3 FFR@640E R"
1520 F$=":O3 CCR@640<B R"
1530 G$=":O2 FFFF FFFE4EFG >DC<BG"
1540  "!" '26,27
1550  RETURN
1560    LABEL "P-A3"
1570 A$="O0 E-E-R4E-E-RE- RE-R4E-E-E-E-"
1580 B$=":O3 R4D4R4D4 R4D4D4DD"
1590 C$=":O3 FFFF FFFE4EFG>DC<BG"
1600 D$=":O6 E2RDC<B4B>CDEDC<A&+"
1610 E$=":O3 FFR@640E Q8 R@896F&+"
1620 F$=":O3 CCR@640<B Q8 R@896>C&+"
1630 G$=":O2 FFFF FFFE4EFG >DC<B Q8 F&+"
1640  "!" '26,28
1650 A$="O0 RE-R4E-E-RE- RE-R4E-E-RE-"
1660 B$=":O3 R4DR4.DR4.DR4.DR"
1670 C$=":O3 FFFF FFFE4EEEEFGA&+"
1680 D$=":O5 A@640GEB@811>C@171C@10D@160"
1690 E$=":O3 F2.RE@640RGRA&+"
1700 F$=":O3 C2.R<B@640R>DRE&+"
1710 G$=":O2 F2.RE@640RGRA&+"
1720  "!" '29,30
1730 C$=":O3 AAAA AAAA AA>E<B >C<BGF
1740 D$=":O6 C@640<B4E@896EG@8A@120&+"
1750 E$=":O3 A1R>DRCR<G4F&+"
1760 F$=":O3 E1RBRARD4C&+"
1770 G$=":O2 A1R>DRCR<G4F&+"
1780  "!" '31,32
1790 C$=":O3 FFFF FFFE4EEE EFGA&+"
1800 D$=":O5 A@640GEB2.>CDC&+"
1810 E$=":O3 F2.RE@640RGRA&+"
1820 F$=":O3 C2.R<B@640R>DRE&+"
1830 G$=":O2 F2.RE@640RGRA&+"
1840  "!" '33,34
1850 C$=":O3 AAAA AAAA AAAA AAAA"
1860 D$=":O6 C<B4A@1280>CEG@8A@120&+"
1870 E$=":O3 A0"
1880 F$=":O3 E0"
1890 G$=":O2 A0"
1900  "!" '35,36
1910 A$="O0 RE-RE-RE-R4E-RE-E-E-E-4R"
1920 B$=":O3 R4DRDR4DRBGDDR4D16D16"
1930 C$=":O3 AAAA AAAA AAAA A>C4."
1940 D$=":O6 A@1792 V111 RC@8D@120&+"
1950 E$=":V110 O4L16I6"+E1$
1960 F$=":O3 AR4AR4AR4.AR4>C4."
1970 G$=":O2 AR4AR4AR4.AR4>C4."
1980  "!" '37,38
1990 A$="O0 E-E-RE-E-4RE- E-E-RE-E-4RE-"
2000 B$=":O3 R4D4R4D4 R4D4R4D4"
2010 C$=":O4 DDDD DDDD DDDD DED<A&+"
2020 D$=":O6 D<A>E<A>F<A>G<A>A<A>B<A>>CC@16D@240C&+"
2030 E$=":V98 O4I3L8 D4RDD4RDRDRDDE4<A"
2040 F$=":O3 A4RAA4RARARAAB4E"
2050 G$=":O3 D4RDD4RDRDRDDE4<A"
2060  "!" '39,40
2070 C$=":O3 AAAA AAGA AAAA A>E<B>C"
2080 D$=":O7 C@640<BAE1C@8D@120&+"
2090 E$=":O3 R4.AA4 RARARAR>C4."
2100 F$=":O3 R4.EE4 RERERERA4."
2110 G$=":O2 AAAAA4 RARARAR>C4."
2120  "!" '41,42
2130    LABEL "P-A4"
2140 C$=":O4 DDDD DDDD DDDD DED<A&+"
2150 D$=":O6 D<A>E<A>F<A>G<A>A<A>B<A>>CC@16D@240C&+"
2160 E$=":O4 D4RDD4RDRDRDDE4<A"
2170 F$=":O3 A4RAA4RARARAAB4E"
2180 G$=":O3 D4RDD4RDRDRDDE4<A"
2190  "!" '39,40
2200 C$=":O3 AAAA AAGA AAGA >DC<BG"
2210 D$=":O7 C@640<BGG@8A@760B>C<B&+"
2220 E$=":O3 R4.AA4 RARARARG4."
2230 F$=":O3 R4.EE4 RERERERD4."
2240 G$=":O2 AAAAA4 RARARARG4."
2250  "!" '43,44
2260 A$="O0 E-E-RE-E-4RE- RE-RE-E-E-RE-"
2270 C$=":O3 FFFF FFFG4GGG GGGE&+"
2280 D$=":O6 B4.RAB>CC@8D@760C<B>D@8E@120&+"
2290 E$=":O3 F2.RG@896RE&+"
2300 F$=":O3 C2.RD@896R<B&+"
2310 G$=":O2 F2.RG@896RE&+"
2320  "!" '45,46
2330 A$="O0 RE-RE-E-4RE- RE-RE-E-E-RE-"
2340 B$=":O3 R4D4R4D4 R4D4R2"
2350 C$=":O3 EEEE GGFE4EEE EG4F&+"
2360 D$=":O7 E@1664DC<B&+"
2370 E$=":O3 ER1.GRF&+"
2380 F$=":O2 BR1.>DC&+"
2390 G$=":O2 EEEE GGFE@640GGRF&+"
2400  "!" '47,48
2410 A$="O0 E-E-RE-E-4RE- RE-RE-E-E-RE-"
2420 B$=":O3 R4D4R4D4 R4D4R4D4"
2430 C$=":O3 FFFF FFFG4GGG GEGA&+"
2440 D$=":O6 B4.RAB>C<B@640RG4G@8A@120&+"
2450 E$=":O3 F2.RG2.R4A&+"
2460 F$=":O3 C2.RD2.R4E&+"
2470 G$=":O2 F2.RG2.EGA&+"
2480  "!" '49,50
2490  RETURN
2500    LABEL "P-A5"
2510 A$="O0 RE-RE-E-4E-4 E-4R4E-4E-4"
2520 B$=":O3 R4D4R4D16D16D16D16 D4R4B32B32B16G32G32G16D4"
2530 C$=":O3 AAAA AAAA AR"
2540 D$=":O6 A@1152R"
2550 E$=":O3 A1AR"
2560 F$=":O3 E1ER"
2570 G$=":O2 A1AR"
2580 H$=":O4 EEEB EEE16E16B16B16 E4R2."
2590  "!" '51,52
2600 RETURN
2610    LABEL "P-B"     ' MATCH'S WEAPON IS BREAD AND OIL
2620 A$="T82 O0 E-4E-4E-4E-4 E-4E-4E-4E-4"
2630 B$=":"
2640 C$=":V115 O3 AAAA AAAA AAAA AAAA"
2650 D$=":V116 I1"+D1$
2660 E$=":" : F$=":" :G$=":" :H$=":"
2670  "!" '53,54
2680  "!" : "!"
2690 B$=":R@1280D4RDDD"
2700 H$=":O4 R V110E V113E V115E V119E V123E V127E V126E V124 E4
E4E4E4"
2710  "!" '55,56
2720 A$="T83 O0 E-4RE-E-4R4 E-E-RE-E-4R"
2730 B$=":O3 R2.D4 R@640GD4"
2740 C$=":V107 O3 F4RFF4R4 F4RFF4F4"
2750 D$=":V110 O6I7 D2.C4 D4.E4.C4"
2760 E$=":V95 O3R4CRFRGR R4CRFRGR"
2770 F$=":O2 R4CRFRGR R4CRFRGR"
2780 H$=":O4 E4E4E4E4 E4E4E4E4"
2790  "!" '57,58
2800 C$=":O3 E4REE4R4 E4REE4G4"
2810 D$=":O6 D4.E@1152R<GA>C"
2820 E$=":O2 R4BR>ERF+R <R4BR>ERF+R"
2830 F$=":O1 R4BR>ERF+R <R4BR>ERF+R"
2840  "!" '59,60
2850 C$=":O3 F4RFF4R4 F4RFF4F4"
2860 D$=":O6 D2.C4D4.E4.C4"
2870 E$=":O3 R4CRFRGR R4CRFRGR"
2880 F$=":O2 R4CRFRGR R4CRFRGR"
2890  "!" '61,62
2900 C$=":O3 E4REE4R4 E4REE4G4"
2910 D$=":O6 D4.C4.<G2.RGA>C"
2920 E$=":O2 R4BR>ERF+R <R4BR>ERF+R"
2930 F$=":O1 R4BR>ERF+R <R4BR>ERF+R"
2940  "!" '63,64
2950 C$=":O3 F4RFF4R4 F4RFF4F4"
2960 D$=":O6 D2.C4 D4.E4.C4"
2970 E$=":O3 R4CRFRGR R4CRFRGR"
2980 F$=":O2 R4CRFRGR R4CRFRGR"
2990  "!" '57,58
3000 C$=":O3 E4REE4R4 E4REE4G4"
3010 D$=":O6 D4.E@1152RCDE"
3020 E$=":O2 R4BR>ERF+R <R4BR>ERF+R"
3030 F$=":O1 R4BR>ERF+R <R4BR>ERF+R"
3040  "!" '59,65
3050 C$=":O3 F4RFF4R4 F4RFF4F4"
3060 D$=":O6 F1G2A2"
3070 E$=":O3 R4CRFRGR R4CRFRGR"
3080 F$=":O2 R4CRFRGR R4CRFRGR"
3090  "!" '66,67
3100 A$="T82 O0 E-4R4E-E-R4 E-4E-4E-4E-4 T81"
3110 B$=":O3 R4D4R4DD L16 RB32B32BBB32B32BBBG32G32GGGDDD8 L8"
3120 C$=":O3 EEEE EEEE EEEE EEEE"
3130 D$=":O6 G+@640EG+BB2.R"
3140 E$=":O2 R4BR>ERF+R <R4BR>ERF+R"
3150 F$=":O1 R4BR>ERF+R <R4BR>ERF+R"
3160 H$=":O4 EEEB EEEB L16 EEEB EEEB EEEB EE L8"
3170  "!" '68,69
3180  RETURN
3190    LABEL "P-B2"    ' MATCH IS POOR AT MAMA LEMON
3200 A$="O0 E-E-R4 E-E-R4 E-E-R4 E-.E-."
3210 B$=":O3 R4D4R4D4 R4D4"+R5$+"D."+L5$+"D."+C5$
3220 C$=":O3 GGGG GGGG GGGG GG"
3230 D$=":O4 G>G-G<G>EG-<G>DE<G>D-D<GB"
3240 E$=":O4 R4I9q8G4RG4FG@496 R@16>I7q5G.G.R"
3250 F$=":O4 R4I9q8D4RD4CD@496 R@16>I7q5D.D.R"
3260 G$=":O4 R1.q5 G.G.R"
3270 H$=":O4 EEEB EEEB EEEB E.E.R"
3280  "!" '76,78
3290 A$="O0 E-4RE-4E-E-"
3300 B$=":O3 R2D4R4.DR4.D"
3310 C$=":O3 EEEE EEEE EEEE EEEE"
3320 D$=":"
3330 E$=":O5 ER@640EE"
3340 F$=":O4 A-R@640A-A-"
3350 G$=":O4 ER@640EE"
3360 H$=":O4 EEEB EEEB EEEB EEEB"
3370  "!" '79,80
3380 A$="O0 E-4RE-4RE-E-"
3390 B$=":O3 RDR2.RDRDRDDD
3400 C$=":O3 EEEE EEEE EEEE EG4."
3410 D$=":V110 R1 O5I7 R@896 A&+"
3420 E$=":O5 RER4ER@640EER"
3430 F$=":O4 RA-R4A-R@640A-A-R"
3440 G$=":O4 RER4ER@640EER"
3450  "!" '81,82
3460  RETURN
3470    LABEL "P-B3"    ' HELP ME! HELP ME! LUCY CRYING
3480 PLAY "::::I3:I3:I3"
3490    LABEL "P-B3'"
3500 A$="O0 RE-R4E-E-RE- RE-R4E-E-RE-"
3510 B$=":O3 R4D4R4D4R4D4R4D4"
3520 C$=":O3 FFFF FFFE4EEEEFGA&+"
3530 D$=":O5 A@640GEB@811>C@171C@10D@160"
```


▶いやーキューティハニーには参りました。しっかりテープに録って友人に聞かせまわっています。

高橋　啓太（19）茨城県

```
4660 KY_2_F_4$="04RC&C4C4CCRC&CCRCRERD&D4C4CDRD&D4C4CDRD&D4"
4670 KY_1_F_5$="04E4EFRF&F4E4EFRF&F4"
4680 KY_2_F_5$="04C4CDRD&D4C4CDRD&D4"
4690 KY_1_F_6$="04E4EFRF&F4E4EFRF&F4E4ERRRE4.E&E2"
4700 KY_2_F_6$="04C4CDRD&D4C4CDRC&C4C4CCRCC4.C&C2"
4710 KY_1_G_1$="04L8CDFFFFFF4F&FECD&D2"
4720 KY_1_G_2$="05L8CCCCCCCCCCDC&C<G4."
4730 KY_2_G_2$="04L8EEEEEEEEEEFE&EC4."
4740 KY_1_G_3$="04B-B-B-B-B-A4B-&B-A4B-&A2A2AB-AG"
4750 KY_2_G_3$="04DDDDDD4D&DD4D&D2C2CDCC"
4760 KY_1_G_4$="04F4.G&G2FFFFFF4F&FECD&D2"
4770 KY_2_G_4$="04C4.C&C2DDDDDD4D&DGCF&F2"
4780 KY_1_G_9$="04V9B-B-B-B-B-A4V8B-&B-A4B-&A2V7A2AB-AG"
4790 KY_2_G_9$="04V9DDDDDD4V8D&DD4D&D2V7C2CDCC"
4800 KY_1_G_10$="04V5F4.G&G2V4FFFFFF4V3F&FECD&D2"
4810 KY_2_G_10$="04V5C4.C&C2V4DDDDDD4V3D&DGCF&F2"
4820 KY_1_G_11$="05L8V2CCCCCCCCV1CCDC&C<G4."
4830 KY_2_G_11$="04L8V2EEEEEEEEV1EEFE&EC4."
4840 '
5000 '     ELECTRIC BASE
5010 '
5020 BA_INT_2$="02L8B->C&C2.<B->C&C4<B->C&C4CD"
5030 BA_INT_5$="R8"+STRING$(29,"D")
5040 BA_INT_6$="<B->C&C2.<B->C&C4<B->C&C4"
5050 BA_A_1$="C<A&A2.R8>E>D<E>C#<EB-<A4>"+"D&D"+STRING$(14,"D")
+"<B-&B-"+STRING$(13,"B-")+"AG&G"+STRING$(14,"G")+"F&FFFFFFB-4"
5060 BA_A_2$="R>CRCRERD&D"+STRING$(14,"D")+"<B-&B-"+STRING$(13,
"B-")+"AG&G"+STRING$(14,"G")+"F&FFFFFAB-B>C4RC&C2"
5070 BA_B_1$="02G2RGRG&G4.G&G2"
5080 BA_B_2$="F4.F&F4GF&F4GF&F2B-2RB-RB-&B-4.>C&C2F4RERDRC&C2"
5090 BA_B_3$="R<B->RC&C2RDRE&EE<E4&E4.G&G4.G#&G#4.A&A1"
5100 BA_C_1$="03"+STRING$(13,"D")+"<AB-B "
5110 BA_C_2$=">CCCCCCCCCCCDC<B-A"
5120 BA_C_3$=STRING$(16,"G")
5130 BA_C_4$=STRING$(8,"F")+">D4.E&ECC4<"
5140 BA_C_5$=STRING$(8,"F")+">D4.E&E4"
5150 BA_D_1$="CDR1R1"
5160 BA_D_2$="02R1R2R4B->C&C2.<B->C&C4<B->C&C4CD"
5170 BA_D_3$=STRING$(3,"R4D4R4D4")
5180 BA_D_4$="R4D4R4<B->C&C2.<B->C&C4<B->C&C4CD"
5190 BA_E_1$="R4E-4&E-FR<G&G"+STRING$(15,"G")
5200 BA_E_2$="03"+STRING$(13,"E-")+"<AB-B"
5210 BA_E_3$="03"+STRING$(15,"C")+"<A"
5220 BA_E_4$=STRING$(13,"B-")+"FRG&GGGG"
5230 BA_E_5$=STRING$(10,"G")+">DE &E-E-E-E-E-E-E-E-"
5240 BA_E_6$="E-E-E-E-E-<AB-B>"+STRING$(16,"C")
5250 BA_E_7$="02BBBBBBBB"
5260 BA_F_1$="F4.FR>CRD&D"+STRING$(13,"D")+"<AB-&B-B-"
5270 BA_F_2$=STRING$(12,"B-")+"AG&GG"
5280 BA_F_3$=STRING$(12,"G")
5290 BA_F_4$="GF&FFFFFFFFR>CRCRERD&D"+STRING$(13,"D")+"<AB-&B-B
-"
5300 BA_F_5$=STRING$(12,"B")+"AGGG"
5310 BA_F_6$=STRING$(13,"G")+"F&FFFFFAB-B>C4RCC2"
5320 BA_G_1$="03C"+STRING$(14,"D")+"<AB-B>"
5330 BA_G_2$=STRING$(12,"C")+"DC<B-A"
5340 BA_G_3$=STRING$(16,"G")+STRING$(8,"F")
5350 BA_G_4$="03D4.E&ECC4"+STRING$(13,"D")+"<AB-B>"
5360 BA_G_9$="V11"+STRING$(8,"G")+"V10"+STRING$(8,"G")+"V8"+STR
ING$(8,"F")
5370 BA_G_10$="03V6D4.E&ECC4"+"V5"+STRING$(8,"D")+"V4"+STRING$(
5,"D")+"<AB-B>"
5380 BA_G_11$="V3"+STRING$(8,"C")+"V2CCCCDC<B-A"
5390 '
6000 '     DRUMS
6010 '
6020 E4$="@36E4"
6030 E8$="@36E8"
6040 A4$="@38A4"
6050 A8$="@38A8"
6060 R4$="R4"
6070 R8$="R8"
6080 PA_1$=A4$+E4$+A4$+E4$
6090 PA_2$=A4$+E4$+A8$+A8$+E4$
6100 '
6110 DR_INT_1$=STRING$(3,PA_1$)+A4$+E4$+A4$
6120 DR_INT_2$=E8$+A8$+R4$+E4$+R4$+E8$+A8$+R4$+E8$+A8$+R4$+E8$+
A8$
6130 DR_INT_3$=R4$+E4$+A4$+E4$+STRING$(2,PA_1$)+A4$+E4$+A4$
6140 DR_INT_5$=R8$+A8$+E4$+A4$+E4$+STRING$(2,PA_1$)+A4$+E4$+A4$
6150 DR_INT_6$=E8$+A8$+R4$+E4$+R4$+E8$+A8$+R4$+E8$+A8$+R4$
6160 DR_A_1$=E8$+A8$+PA_1$+"L8@36GDFDDCR"+A8$+R8$+A8$+E4$+A4$+E
4$+A4$+E4$+A4$+E8$+A8$+R4$+E4$+A4$+E4$+A4$+E4$+A4$+E8$+A8$+R4$+E
4$+A4$+E4$+A4$+E4$+A4$+E8$+A8$+R4$+E4$+A8$+A8$+E4$
6170 DR_A_2_1$=R8$+A8$+E8$+A8$+R8$+A8$+E8$+A8$+R8$+A8$+E4$+A4$+
E4$+A4$+E4$+A4$+E8$+A8$+R4$+E4$+A4$+E4$+A4$+E4$+A4$+E8$+A8$
6180 DR_A_2_2$=R4$+E4$+A4$+E4$+A4$+E4$+A4$+E8$+A8$+R4$+E4$+R8$+
"@36C16C16"+E8$+R8$+A4$+R8$+A8$+R8$+A8$+E4$
6190 DR_B_1$=A4$+E4$+R8$+A8$+E8$+A8$+R4$+E8$+A8$+R4$+E4$
6200 DR_B_2$=A4$+E8$+A8$+R4$+E8$+A8$+R4$+E8$+A8$+R4$+E4$+A4$+E4
$+R8$+A8$+E8$+A8$+R4$+E8$+A8$+R8$+A8$+E4$+A4$+E8$+A8$+R8$+A8$+R8
$+A8$+R8$+A8$+E4$
6210 DR_B_3$=R8$+A8$+R8$+A8$+STRING$(2,R8$+A8$+E4$+R8$+A8$+R8$+
A8$)+R4$+E8$+A8$+R4$+E8$+A8$+R8$+E8$+"E16E8.E8E16E16E16E16E8"
6220 DR_C_1$=PA_1$+PA_1$
6230 DR_C_2$=PA_1$+PA_1$
6240 DR_C_3$=PA_1$+PA_1$
6250 DR_C_4$=A4$+E4$+A8$+E8$+E4$+A4$+E8$+A8$+R8$+A8$+E4$
6260 DR_C_5$=A4$+E4$+A8$+"@36{CFD}8C8C8"+A4$+E8$+A8$+R4$
6270 DR_D_1$=A4$+E8$+A8$+R4$
6280 DR_S_D_2$="@36L804R1R2R4EER2R4EER4EER4EEL4"
6290 DR_B_D_2$="@38@V120L404"+STRING$(16,"A")
6300 DR_S_D_3$=STRING$(6,"RC")
6310 DR_B_D_3$=STRING$(12,"A")
6320 DR_S_D_4$="RCRRL8@37@V120GGFFDDCCGGFFL4{GGE}4{EDD}4"
6330 DR_B_D_4$=STRING$(12,"A")
6340 DR_S_E_1$="@V117{CCC}4@36RERRERERERE"
6350 DR_B_E_1$="@V117AARR8A8R8A16A16RARARA8A8R"
6360 DR_E_2$=PA_1$+PA_1$
6370 DR_E_3$=PA_1$+PA_2$
6380 DR_E_4$=PA_2$+A4$+E4$+R8$+A8$+E8$+A8$+R8$+A8$+E4$
6390 DR_E_5$=A4$+E4$+PA_2$+PA_1$
6400 DR_E_6$=PA_2$+PA_1$+PA_2$
6410 DR_E_7$=PA_2$
6420 DR_F_1$=A8$+A8$+"@36G8C8E8"+A8$+R8$+A8$+R8$+A8$+E4$+A4$+E4
$+PA_2$+A4$
6430 DR_F_2$=E4$+A4$+E4$+PA_2$+A4$
6440 DR_F_3$=E4$+A4$+E4$+A4$+E4$+A8$+A8$
6450 DR_F_4$=E4$+PA_2$+R8$+A8$+E8$+A8$+R8$+A8$+E8$+A8$+R8$+A8$+
E4$+A4$+E4$+PA_2$+A4$
6460 DR_F_5$=E4$+A4$+E4$+PA_2$+A4$
6470 DR_F_6$=E4$+A4$+E4$+PA_2$+PA_2$+A4$+E8$+A8$+R8$+"@36G8G4"
6480 DR_G_1$=A8$+E8$+PA_2$+PA_2$
6490 DR_G_2$=PA_2$+PA_2$
6500 DR_G_3$=PA_2$+PA_2$+PA_2$
6510 DR_G_4$=A4$+E8$+A8$+R4$+E4$+PA_2$+PA_2$
6520 DR_G_9$="@V110"+PA_2$+"@V100"+PA_2$+"@V95"+PA_2$
6530 DR_G_10$="@V88"+A4$+E8$+A8$+R4$+E4$+"@V82"+PA_2$+"@V72"+PA
_2$
6540 DR_G_11$="@V48"+PA_2$+"@V30"+PA_2$
6550 '
10000 '     MAIN
10010 '
10020 PLAY INIT
10030 PLAY "@42L804t165@V105","@171804t165@V105","L804T165@V117"
,"L804T165V12","L804T165V10","L804T165V10"
10040 PLAY EG_INT_1$,KY_1_INT_1$,DR_INT_1$
10050 PLAY EG_INT_2$,KY_1_INT_2$,DR_INT_2$,BA_INT_2$,KY_2_INT_2$
,KY_3_INT_2$
10060 PLAY EG_INT_3$,KY_1_INT_1$,DR_INT_3$
10070 PLAY EG_INT_2$,KY_1_INT_2$,DR_INT_2$,BA_INT_2$,KY_2_INT_2$
,KY_3_INT_2$
10080 PLAY EG_INT_5$,KY_1_INT_1$,DR_INT_5$,BA_INT_5$,KY_2_INT_5$
,KY_3_INT_5$
10090 PLAY EG_INT_6$,KY_1_INT_6$,DR_INT_6$,BA_INT_6$,KY_2_INT_6$
,KY_3_INT_6$
10100 PLAY EG_A_1$,"@44@V110"+LY_A_1$,DR_A_1$,BA_A_1$,KY_1_A_1$,
KY_2_A_1$
10110 PLAY EG_A_2$,LY_A_2$,DR_A_2_1$+DR_A_2_2$,BA_A_2$,KY_1_A_2$
,KY_2_A_2$
10120 PLAY EG_B_1$,LY_B_1$,DR_B_1$,BA_B_1$,KY_1_B_1$,KY_2_B_1$
10130 PLAY EG_B_2$,LY_B_2$,DR_B_2$,BA_B_2$,KY_1_B_2$,KY_2_B_2$
10140 PLAY EG_1_B_3$+KY_3_B_3$+EG_3_B_3$,LY_B_3$,DR_B_3$,BA_B_3$
,KY_1_B_3$+EG_2_B_3$,KY_2_B_3$
10150 PLAY EG_C_1$,LY_C_1$,DR_C_1$,BA_C_1$,KY_1_C_1$,KY_2_C_1$
10160 PLAY EG_C_2$,LY_C_2$,DR_C_2$,BA_C_2$,KY_1_C_2$,KY_2_C_2$
10170 PLAY EG_C_3$,LY_C_3$,DR_C_3$,BA_C_3$,KY_1_C_3$,KY_2_C_3$
10180 PLAY EG_C_4$,LY_C_4$,DR_C_4$,BA_C_4$,KY_1_C_4$,KY_2_C_4$
10190 PLAY EG_C_1$,LY_C_1$,DR_C_1$,BA_C_1$,KY_1_C_1$,KY_2_C_1$
10200 PLAY EG_C_2$,LY_C_2$,DR_C_2$,BA_C_2$,KY_1_C_2$,KY_2_C_2$
10210 PLAY EG_C_3$,LY_C_3$,DR_C_3$,BA_C_3$,KY_1_C_3$,KY_2_C_3$
10220 PLAY EG_C_5$,LY_C_5$,DR_C_5$,BA_C_5$,KY_1_C_5$,KY_2_C_5$
10230 PLAY EG_D_1$,LY_D_1$,DR_D_1$,BA_D_1$,KY_1_D_1$,KY_2_D_1$+E
G_2_D_1$
10240 PLAY EG_D_2$,DR_S_D_2$,DR_B_D_2$,BA_D_2$,KY_1_D_2$,KY_2_D_
2$
10250 PLAY EG_D_3$,DR_S_D_3$,DR_B_D_3$,BA_D_3$,KY_1_D_3$
10260 PLAY EG_D_4$,DR_S_D_4$,DR_B_D_4$,BA_D_4$,KY_1_D_4$,KY_2_D_
4$
10270 PLAY "@V110"+EG_E_1$,DR_S_E_1$,DR_B_E_4$,BA_E_1$,KY_1_E_1$
,KY_2_E_1$
10280 PLAY EG_E_2$,"@17V10"+KY_1_E_2$,DR_E_2$,BA_E_2$,KY_2_E_2$,
KY_3_E_2$
10290 PLAY EG_E_3$,KY_1_E_3$,DR_E_3$,BA_E_3$,KY_2_E_3$,KY_3_E_3$
10300 PLAY EG_E_4$,KY_1_E_4$,DR_E_4$,BA_E_4$,KY_2_E_4$,KY_3_E_4$
10310 PLAY EG_E_5$,KY_1_E_5$,DR_E_5$,BA_E_5$,KY_2_E_5$,KY_3_E_5$
10320 PLAY EG_E_6$,KY_1_E_6$,DR_E_6$,BA_E_6$,KY_2_E_6$,KY_3_E_6$
10330 PLAY EG_E_7$,KY_1_E_7$,DR_E_7$,BA_E_7$,KY_2_E_7$,KY_3_E_7$
10340 PLAY EG_F_1$,"@44@V110"+LY_F_1$,DR_F_1$,BA_F_1$,KY_1_F_1$,
KY_2_F_1$
10350 PLAY EG_F_2$,LY_F_2$,DR_F_2$,BA_F_2$,KY_1_F_2$,KY_2_F_2$
10360 PLAY EG_F_3$,LY_F_3$,DR_F_3$,BA_F_3$,KY_1_F_3$,KY_2_F_3$
10370 PLAY EG_F_4$,LY_F_4$,DR_F_4$,BA_F_4$,KY_1_F_4$,KY_2_F_4$
10380 PLAY EG_F_5$,LY_F_5$,DR_F_5$,BA_F_5$,KY_1_F_5$,KY_2_F_5$
10390 PLAY EG_F_6$,LY_F_6$,DR_F_6$,BA_F_6$,KY_1_F_6$,KY_2_F_6$
10400 PLAY EG_B_1$,LY_B_1$,DR_B_1$,BA_B_1$,KY_1_B_1$,KY_2_B_1$
10410 PLAY EG_B_2$,LY_B_2$,DR_B_2$,BA_B_2$,KY_1_B_2$,KY_2_B_2$
10420 PLAY EG_1_B_3$+KY_3_B_3$+EG_3_B_3$,LY_B_3$,DR_B_3$,BA_B_3$
,KY_1_B_3$+EG_2_B_3$,KY_2_B_3$
10430 PLAY EG_C_1$,LY_C_1$,DR_C_1$,BA_C_1$,KY_1_C_1$,KY_2_C_1$
10440 PLAY EG_C_2$,LY_C_2$,DR_C_2$,BA_C_2$,KY_1_C_2$,KY_2_C_2$
10450 PLAY EG_C_3$,LY_C_3$,DR_C_3$,BA_C_3$,KY_1_C_3$,KY_2_C_3$
10460 PLAY EG_C_4$,LY_C_4$,DR_C_4$,BA_C_4$,KY_1_C_4$,KY_2_C_4$
10470 PLAY EG_C_1$,LY_C_1$,DR_C_1$,BA_C_1$,KY_1_C_1$,KY_2_C_1$
10480 PLAY EG_C_2$,LY_C_2$,DR_C_2$,BA_C_2$,KY_1_C_2$,KY_2_C_2$
10490 PLAY EG_C_3$,LY_C_3$,DR_C_3$,BA_C_3$,KY_1_C_3$,KY_2_C_3$
10500 PLAY EG_C_5$,LY_C_5$,DR_C_5$,BA_C_5$,KY_1_C_5$,KY_2_C_5$
10510 PLAY EG_G_1$,LY_G_1$,DR_G_1$,BA_G_1$,KY_1_G_1$,EG_2_G_1$
10520 PLAY EG_G_2$,LY_G_2$,DR_G_2$,BA_G_2$,KY_1_G_2$,KY_2_G_2$
10530 PLAY EG_G_3$,LY_G_3$,DR_G_3$,BA_G_3$,KY_1_G_3$,KY_2_G_3$
10540 PLAY EG_G_4$,LY_G_4$,DR_G_4$,BA_G_4$,KY_1_G_4$,KY_2_G_4$
10550 PLAY EG_G_2$,LY_G_5$,DR_G_2$,BA_G_2$,KY_1_G_2$,KY_2_G_2$
10560 PLAY EG_G_3$,LY_G_6$,DR_G_3$,BA_G_3$,KY_1_G_3$,KY_2_G_3$
10570 PLAY EG_G_4$,LY_G_7$,DR_G_4$,BA_G_4$,KY_1_G_4$,KY_2_G_4$
10580 PLAY EG_G_2$,LY_G_8$,DR_G_2$,BA_G_2$,KY_1_G_2$,KY_2_G_2$
10590 PLAY EG_G_9$,LY_G_9$,DR_G_9$,BA_G_9$,KY_1_G_9$,KY_2_G_9$
10600 PLAY EG_G_10$,LY_G_10$,DR_G_10$,BA_G_10$,KY_1_G_10$,KY_2_G
_10$
10610 PLAY EG_G_11$,LY_G_11$,DR_G_11$,BA_G_11$,KY_1_G_11$,KY_2_G
_11$
```

▶おい６月号の21ページに載っていた（で）とかいう奴。なにが「まじゃべんちゃーのうさぎちゃんは私のものだ」だ。私はずーっと以前から目をつけてたんでぃ！　おまえなんかにやるもんか。僕のものだい！　悔しかったらどっちがロリコンと呼ばれているか勝負してみるかい？　ふっふっふっ。

一戸　譲（17）宮城県

SHORT ACCESS

最近はショートプログラムの投稿も増えてきて，まことに喜ばしい限り。今月はなんとMZ-700用の高速LINEルーチンとX1 turboでX1用の漢字ROMを使用したアプリケーションを実行するためのBASIC改造プログラムです。

MZ-700用
超高速(?)LINEルーチン

Takada Masami
高田 正実

LINEルーチンというと以前，試験に出るX1でアルゴリズムが紹介されたことがありますが，これはそういったものや雑誌でよく紹介されているものとはまったく異なるアルゴリズムによるものです(Oh!FM誌には以前載っていた)。MZ-700にはこういったアルゴリズムのほうが，ほかのものに比べてはるかに有利なのです。たぶんX68000でもそうでしょう。

非常に短いソースリストですので特にアルゴリズムの解説はしません。暇と興味のある人はぜひとも解析してみてください。

入力方法

モニタからC800$_H$～C92C$_H$までを入力してください。

＊JC800

で起動すると，てきとーに線を引き始めます（キャラクタ単位なのであまり期待しないように)。

このときファンクションキーを押してリターンキーを押すと右から左に線を引きます。ただし，速すぎてほとんど識別できませんが。パラメータは，

LINE(X1, Y1)－(X2, Y2)

の場合,

(IX)＝Y1　　(IX＋1)＝X1
(IX＋2)＝Y2　　(IX＋3)＝X2

となっています。

Profile
◇高田さんは群馬県にお住まいの19歳，現在大学2年生です。使用機種はMZ-700とMZ-2500V2。5月の投稿数19本という記録の持ち主です。

リスト1 LINEルーチン

```
C800 21 00 D0 11 01 D0 01 E8 : BC
C808 03 36 5A ED B0 21 00 D8 : 29
C810 11 01 D8 01 E8 03 75 ED : 38
C818 B0 3E 09 32 00 E0 3A 01 : 44
C820 E0 3C C2 7F C8 DD 21 00 : 23
C828 CF ED 5F E6 77 32 FF CF : 78
C830 CD 66 C8 DD 77 00 CD 4D : 69
C838 C8 DD 77 01 CD 66 C8 DD : F5
C840 77 02 CD 4D C8 DD 77 03 : B2
C848 CD A2 C8 18 D8 ED 5F FD : 70
C850 86 00 E6 1F 4F ED 5F FD : 23
C858 86 FF E6 07 81 4F ED 5F : 8E
C860 E6 01 81 FD 23 C9 ED 5F : 9D
C868 FD 86 00 E6 0F 4F ED 5F : 13
C870 FD 86 09 E6 07 81 4F ED : 36
C878 5F E6 01 81 FD 23 C9 DD : 8D
---------------------------------
SUM: B8 77 57 49 C2 0B 79 8B D56F
```

```
C880 21 00 CF DD 36 00 00 DD : E0
C888 36 02 18 21 FF CF 34 06 : 79
C890 28 C5 78 3D DD 77 01 DD : D4
C898 77 03 CD A2 C8 C1 10 F1 : 73
C8A0 18 E9 DD 7E 03 DD 96 01 : D3
C8A8 0E 00 CA BA C8 0E 23 D2 : 5D
C8B0 BA C8 DD 7E 01 DD 96 03 : 54
C8B8 0E 2B 3C 57 47 79 32 29 : E7
C8C0 C9 21 00 00 4D DD 7E 02 : 94
C8C8 DD 96 00 CA DD C8 21 28 : 2B
C8D0 00 D2 DD C8 DD 7E 00 DD : AF
C8D8 96 02 21 D8 FF 3C 5F 22 : 4D
C8E0 18 C9 22 25 C9 DD 6E 00 : 3C
C8E8 C5 D5 DD 5E 01 16 00 62 : 4E
C8F0 29 29 29 44 4D 29 29 09 : 67
C8F8 19 01 00 D8 09 D1 C1 3A : C7
---------------------------------
SUM: 3F F9 12 F3 13 94 1C 7E 0D4F
```

```
C900 FF CF 32 09 C9 32 1D C9 : EA
C908 36 00 79 83 4F 79 92 CA : 56
C910 21 C9 DA 29 C9 4F C5 01 : CB
C918 00 00 09 C1 36 00 C3 0D : D0
C920 C9 0E 00 C5 01 00 00 09 : A6
C928 C1 00 10 DC C9 00 FF FF : 74
C930 00 00 FF FF 00 00 FF FF : FC
C938 00 00 FF FF 00 00 FF FF : FC
C940 00 00 FF FF 00 00 FF FF : FC
C948 00 00 FF FF 00 00 FF FF : FC
C950 00 00 FF FF 00 00 FF FF : FC
C958 00 00 FF FF 00 00 FF FF : FC
C960 00 00 FF FF 00 00 FF FF : FC
C968 00 00 FF FF 00 00 FF FF : FC
C970 00 00 FF FF 00 00 FF FF : FC
C978 00 00 FF FF 00 00 FF FF : FC
---------------------------------
SUM: E0 A6 94 0D E1 FA 2C 9F BF3E
```

リスト2 LINEルーチンソースリスト

```
C800                  ORG $C800
C800           COLOR  EQU $CFFF
C800           MAIN
C800 21 00 D0         LD HL,$D000
C803 11 01 D0         LD DE,$D001
C806 01 E8 03         LD BC,1000
C809 36 5A            LD (HL),$5A
C80B ED B0            LDIR
C80D 21 00 D8         LD HL,$D800
C810 11 01 D8         LD DE,$D801
C813 01 E8 03         LD BC,1000
C816 75               LD (HL),L
C817 ED B0            LDIR
C819 3E 09            LD A,9
C81B 32 00 E0         LD ($E000),A
C81E 3A 01 E0         LD A,($E001)
C821 3C               INC A
C822 C2 7F C8         JP NZ,YOKO
C825           RND
C825 DD 21 00 CF      LD IX,$CF00
C829 ED 5F            LD A,R
C82B E6 77            AND $77
C82D 32 FF CF         LD (COLOR),A
C830 CD 66 C8         CALL HANGON
C833 DD 77 00         LD (IX),A
C836 CD 4D C8         CALL OUTRUN
C839 DD 77 01         LD (IX+1),A
C83C CD 66 C8         CALL HANGON
C83F DD 77 02         LD (IX+2),A
C842 CD 4D C8         CALL OUTRUN
C845 DD 77 03         LD (IX+3),A
C848 CD A2 C8         CALL LINE
C84B 18 D8            JR RND
C84D           OUTRUN
C84D ED 5F            LD A,R
C84F FD 86 00         ADD A,(IY)
C852 E6 1F            AND 31
C854 4F               LD C,A
C855 ED 5F            LD A,R
C857 FD 86 FF         ADD A,(IY-1)
C85A E6 07            AND 7
C85C 81               ADD A,C
C85D 4F               LD C,A
C85E ED 5F            LD A,R
C860 E6 01            AND 1
C862 81               ADD A,C
C863 FD 23            INC IY
C865 C9               RET
C866           HANGON
C866 ED 5F            LD A,R
C868 FD 86 00         ADD A,(IY)
C86B E6 0F            AND 15
C86D 4F               LD C,A
C86E ED 5F            LD A,R
C870 FD 86 09         ADD A,(IY+9)
C873 E6 07            AND 7
C875 81               ADD A,C
C876 4F               LD C,A
C877 ED 5F            LD A,R
C879 E6 01            AND 1
C87B 81               ADD A,C
C87C FD 23            INC IY
C87E C9               RET
C87F           YOKO
C87F DD 21 00 CF      LD IX,$CF00
C883 DD 36 00 00      LD (IX),0
C887 DD 36 02 18      LD (IX+2),24
C88B           YX
C88B 21 FF CF         LD HL,COLOR
C88E 34               INC (HL)
C88F           ;
C88F 06 28            LD B,40
C891           YK
C891 C5               PUSH BC
C892 78               LD A,B
C893 3D               DEC A
C894 DD 77 01         LD (IX+1),A
C897 DD 77 03         LD (IX+3),A
C89A CD A2 C8         CALL LINE
C89D C1               POP BC
```

```
C89E 10 F1              DJNZ YK
C8A0 18 E9              JR YX
C8A2                LINE
C8A2                ;LD IX,$CF00
C8A2 DD 7E 03           LD A,(IX+3)
C8A5 DD 96 01           SUB (IX+1)
C8A8 0E 00              LD C,0
C8AA CA BA C8           JP Z,STX1
C8AD 0E 23              LD C,$23
C8AF D2 BA C8           JP NC,STX1
C8B2 DD 7E 01           LD A,(IX+1)
C8B5 DD 96 03           SUB (IX+3)
C8B8 0E 2B              LD C,$2B
C8BA                STX1
C8BA 3C                 INC A
C8BB 57                 LD D,A
C8BC 47                 LD B,A
C8BD 79                 LD A,C
C8BE 32 29 C9           LD (LP2),A
C8C1 21 00 00           LD HL,0
C8C4 4D                 LD C,L
C8C5 DD 7E 02           LD A,(IX+2)
C8C8 DD 96 00           SUB (IX)
C8CB CA DD C8           JP Z,STX2
C8CE 21 28 00           LD HL,40
C8D1 D2 DD C8           JP NC,STX2
C8D4 DD 7E 00           LD A,(IX)
C8D7 DD 96 02           SUB (IX+2)
C8DA 21 D8 FF           LD HL,-40
C8DD                STX2
C8DD 3C                 INC A
C8DE 5F                 LD E,A
C8DF 22 18 C9           LD (UYP1+1),HL
C8E2 22 25 C9           LD (UYP2+1),HL
C8E5 DD 6E 00           LD L,(IX)
C8E8
C8E8                ;
C8E8                TDXY
C8E8 C5                 PUSH BC
C8E9 D5                 PUSH DE
C8EA DD 5E 01           LD E,(IX+1)
C8ED 16 00              LD D,0
C8EF 62                 LD H,D
C8F0 29                 ADD HL,HL
C8F1 29                 ADD HL.HL
C8F2 29                 ADD HL,HL
C8F3 44 4D              LD BC,HL
C8F5 29                 ADD HL,HL
C8F6 29                 ADD HL,HL
C8F7 09                 ADD HL,BC
C8F8 19                 ADD HL,DE
C8F9 01 00 D8           LD BC,$D800
C8FC 09                 ADD HL,BC
C8FD D1                 POP DE
C8FE C1                 POP BC
C8FF
C8FF 3A FF CF           LD A,(COLOR)
C902 32 09 C9           LD (PS1+1),A
C905 32 1D C9           LD (PS2+1),A
C908                ;
C908                LP
C908                PS1
C908 36 00              LD (HL),0
C90A 79                 LD A,C
C90B 83                 ADD A,E
C90C 4F                 LD C,A
C90D                LPC
C90D 79                 LD A,C
C90E 92                 SUB D
C90F CA 21 C9           JP Z,LP1
C912 DA 29 C9           JP C,LP2
C915 4F                 LD C,A
C916 C5                 PUSH BC
C917 01 00 00       UYP1 LD BC,0
C91A 09                 ADD HL,BC
C91B C1                 POP BC
C91C                PS2
C91C 36 00              LD (HL),0
C91E C3 0D C9           JP LPC
C921                LP1
C921 0E 00              LD C,0
C923 C5                 PUSH BC
C924 01 00 00       UYP2 LD BC,0
C927 09                 ADD HL,BC
C928 C1                 POP BC
C929                LP2
C929 00                 NOP
C92A 10 DC              DJNZ LP
C92C C9                 RET
```

X1turbo/Z用
X1用漢字ROM対応BASIC

Toyota Kazuki
豊田 和紀

私は以前X1Fを愛用していました。ディスクドライブ/漢字ROM内蔵でもれなくNEW BASICがついてくる。当然それらを利用したプログラムをたくさん作りました。そしてある日，私はX1turboを手に入れました。X1の正しきユーザーである私は当然X1時代に開発したプログラムを使おうと思います。ところが，そのとき悲劇は訪れたのです。

「か，漢字が四角に化けた……」

賢明な読者ならすでにおわかりでしょう。そう，X1とX1turboでは漢字ROMのアドレスが違うのです。X1turboでCZ-8FB01を走らせた場合，

POSITION X, Y

PATTERN −16, KANJI$(CODE)

としても正しい表示はされません。この点が唯一X1turboのアッパーコンパチでないところといえるでしょう。

そこで私はノーマルのX1turboでもCZ-8FB01で正しく漢字が表示されるようにしてみました。漢字表示だけでは芸がないので，高解像度対応，PCG高速アクセスモード対応もつけておきました。

入力&使用法

プログラムはすべてBASICで書かれていますので，BASICからそのとおりに打ち込んでください。特に16進数のマシン語データには十分に注意してください。打ち終わったら忘れずにセーブしておきましょう。

実行方法です。まず，FORMAT & SYSGENで新しいディスクをCZ-8FB01のシステムディスクとします(NEW BASIC可)。そして，先ほど打ち込んだプログラムを実行してください。あとは指示に従ってBASICを書き換えるだけです。“Complete”と表示されたら書き換え完了。書き換えた機能がちゃんと動くことを確認するまでは大事なディスクは入れないように注意してください。

なお，当然のことながらこの書き換えはX1turbo/Zでのみ有効です。X1の人は使わないでください。

Profile

◇豊田さんは新潟県にお住まいの21歳，大学4年生です。マイコン歴は約5年，現在はX1turbo Zのユーザーです。

リスト3 X1漢字ROM対応BASIC修正部

```
10 INIT:WIDTH 80:SCREEN:DEFINT A-Z:CLEAR &HEFFF:BF=&HF000
20 '
30 CLS:LOCATE 0,0:PRINT "'BASIC CZ-8FB01' ﾉ ﾊｲｯﾀ Disk ｦ Drive 1 ﾆ ｲﾚﾃｸﾀﾞｻｲ｡"
40 LOCATE 0,2:PRINT "HIT SPACE KEY"
50 A$=INKEY$:IF A$=" " ELSE 50
60 RC=16:"READ":F$=MEM$(BF+1,16)
70 IF LEFT$(F$,13)="BASIC CZ8FB01" ELSE LOCATE 0,4:PRINT "ｺﾉ Disk ﾆﾊ 'BASIC' ｶﾞ
ｱﾘﾏｾﾝ｡":BEEP:END
80 VR=VAL(RIGHT$(F$,2))/10:ON VR RESTORE "V10","V20"
90 '
100 RC=&H20:"READ"
110 READ M$:MEM$(BF+1,2)=HEXCHR$(M$):MEM$(BF+&H34,2)=HEXCHR$("C0 0A")
120 "WRITE"
130 '
140 '   PCG ROUTINE & KANJI READ SUB
150 '
160 RC=&H2A:"READ"
170 MEM$(BF+&HAB, 5)=HEXCHR$("D5 D9 C5 D5 E5")
180 MEM$(BF+&HB0,16)=HEXCHR$("CD DD 0A D9 04 ED A3 0C 0C 15 C2 B4 0A 18 14 00")
190 MEM$(BF+&HC0,16)=HEXCHR$("C5 D5 D9 C5 D5 E5 CD DD 0A D9 ED A2 04 0C 0C 15")
200 MEM$(BF+&HD0,16)=HEXCHR$("C2 CA 0A FB D9 E1 D1 C1 D9 D1 C1 C9 00 D9 E5 01")
210 MEM$(BF+&HE0,16)=HEXCHR$("D0 1F 3E 20 ED 79 01 FF 3F AF ED 79 06 37 ED 51")
220 MEM$(BF+&HF0,16)=HEXCHR$("7B FE 14 26 00 28 02 26 20 06 27 ED 61 43 0E 00")
230 "WRITE":RC=RC+1:"READ"
```

```
240 MEM$(BF     ,16)=HEXCHR$("16 08 E1 D9 F3 C9 00 01 00 14 16 10 ED A2 04 0C")
250 MEM$(BF+&H10,16)=HEXCHR$("15 20 F9 C9 00 00 00 00 00 00 00 00 00 CD 1B 00")
260 MEM$(BF+&H20, 6)=HEXCHR$("C5 06 1D ED 41 C9")
270 "WRITE"
280 '
290 '   KANJI READ ROUTINE
300 '
310 READ R$,A$:RC=VAL("&H"+R$):"READ":OF=BF+VAL("&H"+A$)
320 MEM$(OF     ,16)=HEXCHR$("C6 20 5F 7D C6 20 57 D5 01 09 00 11 14 0B 21 18")
330 MEM$(OF+&H10,16)=HEXCHR$("00 ED B0 01 09 00 11 18 00 21 1D 0B ED B0 D1 01")
340 MEM$(OF+&H20,16)=HEXCHR$("B6 2F DF 30 03 11 00 80 01 D0 1F 3A E3 0A C6 40")
350 MEM$(OF+&H30,16)=HEXCHR$("ED 79 01 FF 37 ED 59 06 27 AF ED 79 06 3F 7A E6")
360 MEM$(OF+&H40,16)=HEXCHR$("9F ED 79 01 09 00 11 18 00 21 14 0B ED B0 E1 CD")
370 MEM$(OF+&H50,12)=HEXCHR$("07 0B 01 FF 3F CB F7 ED 79 CD 07 0B")
380 "WRITE"
390 '
400 '   START UP ROUTINE (CRTC DATA)
410 '
420 READ R$,A$:RC=VAL("&H"+R$):"READ":OF=BF+VAL("&H"+A$)
430 READ M1$,M2$
440 MEM$(OF     ,16)=HEXCHR$("31 00 00 01 F0 1F ED 78 E6 01 20 10 21"+M1$+"11")
450 MEM$(OF+&H10,16)=HEXCHR$("DD 00 01 14 00 ED B0 3E 23 32 E3 0A 21"+M2$+"11")
460 MEM$(OF+&H20,16)=HEXCHR$("01 00 01 02 00 ED B0 C3 FA 00 35 28 2D 84 1B 00")
470 MEM$(OF+&H30,14)=HEXCHR$("19 1A 00 0F 00 00 00 00 00 00 6B 50 59 88")
480 "WRITE"
490 '
500 IF VR=1 THEN 630
510 '
520 '   NEW BASIC PCG PATCH
530 '
540 RC=&H9C:"READ"
550 MEM$(BF+&H1A, 6)=HEXCHR$("1E 15 CD 33 00 1C")
560 MEM$(BF+&H20,16)=HEXCHR$("CD 33 00 1C CD 33 00 C9 00 00 00 00 00 00 00 00")
570 MEM$(BF+&H30,16)=HEXCHR$("00 00 00 00 00 00 00 00 00 00 00 00 1E 15 CD 2B")
580 MEM$(BF+&H40,16)=HEXCHR$("00 1C CD 2B 00 1C CD 2B 00 C9 00 00 00 00 00 00")
590 MEM$(BF+&H50,16)=HEXCHR$("00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00")
600 MEM$(BF+&H60, 9)=HEXCHR$("00 00 00 00 00 00 00 00 00")
610 "WRITE"
620 '
630 LOCATE 0,4:PRINT "Completed !":CLEAR &HFEFF:END
640 '
650 LABEL "V10"
660 DATA 80 A7,A6,94,C7,80,AA A7,A8 A7
670 '
680 LABEL "V20"
690 DATA 00 BF,91,0D,DF,00,2A BF,28 BF
700 '
710 LABEL "READ"
720   DEVI$ "1:",RC,D1$,D2$
730   MEM$(BF,128)=D1$:MEM$(BF+128,128)=D2$
740 RETURN
750 '
760 LABEL "WRITE"
770   D1$=MEM$(BF,128):D2$=MEM$(BF+128,128)
780   DEVO$ "1:",RC,D1$,D2$
790 RETURN
```

連載　構造化言語SLANG入門(2)

第67部　マルチウィンドウドライバMW-1

●SLANG入門後編

今回はSLANGの配列，間接変数などを取り扱います。ポインタの考え方など，BASICしか扱ったことのない方にはわかりにくい部分もあるかもしれません。しかし，こういった概念を会得することは，あらゆるプログラム言語を扱ううえでも参考になるでしょう。

そのほか，SLANGの基本的な部分はC言語と似たようなものですので，今月の特集記事を参考にしてみてください。

●ウィンドウパッケージ

マルチウィンドウはWALRUSでも使われていましたが，今回は汎用に作られたパッケージです。が，完全なマルチウィンドウ処理というわけではなく，いわばポップアップウィンドウといった感じのものです。特定キーを指定してやればキー入力時にアルゴ機能モドキを作ることも簡単でしょう。MZ-2500のようにメモリに余裕はありませんから，プログラムを常駐させることは難しそうですが，MEMAXを書き換えてメモリを確保しておきメニューに応じてデバイスから読み込むというのが妥当なところでしょうか。このほかにもこのパッケージを使用したアプリケーションがいろいろと考えられるでしょう。トランジェントコマンドのような扱いも面白いかもしれません。

メモリの関係上，仮想画面上の情報保護といったことは現実的ではありませんから，こういった感じのものになるのはやむをえないでしょう。なかなかしっかり作られていますが，まだ未開拓の分野ですので改良すべき点もあるかもしれません。皆さんの意見をお待ちします。

●なぜか突然アルギース

さて，今月のゲームレビューでも取りあげられている「アルギースの翼」ですが，エンディングにOh! MZの文字があるのを不思議に思った方がいるかもしれません（まだ解き終わってないかもしれませんが）。このゲームの移植はなんとS-OS“SWORD”上でZEDAとE-MATEを使って行われています。CP/Mよりも使いやすかったというのが理由のようですが，なんとも驚きですね。ボード上のCTCを判別してCTC搭載ボードがあればノーマルX1でも動作したり，背景がPCG化され88版よりもチラツキのない画面になっていたりと，なかなかよい移植がされているようです。まずはひと安心かな。

全機種共通システム掲載記事

＊以上のアプリケーションは，基本システムであるS-OS“MACE”またはS-OS“SWORD”がないと動作しませんのでご注意ください。

全機種共通S-OS“SWORD”要

構造化言語SLANG入門(2)

配列と間接変数を使う

今回はC言語ライクな配列と間接変数の使用法を中心により高度な使い方やSLANGの拡張法などを見てみましょう。残念ながらSLANG入門は今回で終了ですが，機会があれば飛び入り講座なども開けるかもしれません。大貫さん，お疲れさまでした。

大貫 信昭 *Ohnuki Nobuaki*

はじめに

今回は，間接変数と配列の扱い方について説明しましょう。

前回説明しましたように，SLANGではなにもかも関数として記述します。局所的な変数や配列を使えば，名前の重複を気にしなくてすみますし，ほかの関数に影響を与えずに仕事だけをしてくれる関数を書くことができます。このような関数は引数になにを与えてやり，どんな仕事をしてくれるかという仕様さえわかっていれば，関数の中身について一切考える必要がありません。これを関数の「ブラックボックス化」といいます。関数を「ブラックボックス化」すると余計なことを考えずにすみプログラミングの負担が減りますし，その関数はライブラリとしてほかのプログラムにもそのまま使えますから，便利なことこのうえありません。SLANGのプログラミングに慣れてきたら，関数の「ブラックボックス化」を心がけてください。

さて，ある関数内の局所的な変数や配列は，ほかの関数からアクセスすることができません。となると少し困ったことが出てきます。たとえば，引数として与えた局所的な配列の要素を参照したり変更したりする関数は書けないことになってしまいます。SLANGでは，文字列を配列として扱い，BASICのように文字列を操作する組み込み関数などありませんから，このような関数が書けないと事実上局所的な配列は使えない，文字列も扱えないということになります。

そこで登場するのが間接変数で，これを使えばこのような関数も簡単に書くことができるのです。間接変数はFuzzyBASICの変数の扱いとほぼ同じで，変数としても配列としても扱えます。たとえば，POINTを1バイト型の間接変数として宣言し，

POINT＝$D000;
VARA＝POINT[0];
VARB＝POINT[1];

のようにすると，VARA，VARBにはそれぞれD000H，D001Hのメモリの内容が代入されます。SLANGには専用の変数や配列がありますので，これはいったいなにに使うのかなと疑問に思った方もいるでしょう。間接変数は引数として渡された配列を関数内でアクセスするのにもっとも威力を発揮します。

配列と間接変数

それではサンプルプログラム，リスト1に沿って間接変数と配列の具体的な使い方を説明していきましょう。リスト1は配列の各要素に4を加え，変更前と変更後を表示するプログラムです。なお入力する際，文カッコの｛｝が使えない機種では［ ］などで代用してください。

まず初めに1バイト型の配列ARY1とARY2を宣言し，同時に配列要素の初期化も行っています。ARY2の要素は3個ありますが，初期値は1個しかありません。この場合，残りの2個は0に初期化されます。

配列名は配列が格納されているアドレスを値として持っています。もちろん変数ではありませんから代入はできませんが，参照することはできます。17行のように，

PRINT(HEX4$(ARY1));

などとすれば，配列ARY1が格納されているアドレスを表示することができるのです。

PRTARYは1バイト型配列を表示する関数ですが，引数として配列のアドレスと要素の数を渡しています。SLANGでは引数として16ビットの値しか扱えませんので，配列をまるごと渡すことができません。ですから代わりに配列が格納されているアドレスを渡してやるわけです。アドレスさえわかれば，関数側でアクセスすることができますが，PRTARYの場合要素の数も必要なので一緒に渡しています。

PRTARYの関数定義を見てください。仮引数が，

リスト1 配列の使い方

```
 1 //
 2 //  配列.SL
 3 //
 4 MAIN()
 5   ARRAY
 6     BYTE ARY1[3]={
 7       0,1,2,3        (* [0],[1],[2],[3] *)
 8     },
 9     BYTE ARY2[2]={
10       4              (* [0],[1],[2]     *)
11     };
12
13   VAR I;
14
15   BEGIN
16     PRINT("ARY1: ");
17     PRINT(HEX4$(ARY1));
18     PRINT(/);
19 //
20     PRTARY(ARY1,3);
21     ADDARY(ARY1,3,4);
22     PRTARY(ARY1,3);
23 //
24     PRINT("ARY2: ");
25     PRINT(HEX4$(ARY2));
26     PRINT(/);
27 //
28     PRTARY(ARY2,2);
29     ADDARY(ARY2,2,4);
30     PRTARY(ARY2,2);
31   END;
32 //
33 //
34 PRTARY( BYTE ARY[], N )
35   VAR I;
36
37   BEGIN
38     FOR I=0 TO N {
39       PRINT("[",I,"] : ");
40       PRINT(ARY[I]);
41       PRINT(/);
42     }
43     PRINT(/);
44   END;
45 //
46 //
47 ADDARY( BYTE ARY[], N, DATA )
48   VAR I;
49
50   BEGIN
51     FOR I=0 TO N {
52       ARY[I]=ARY[I]+DATA;
53     }
54   END;
55 //
56 //  END
57 //
```

リスト2　2次元配列の例

```
 1 //
 2 //   2次元配列.SL
 3 //
 4 MAIN()
 5   ARRAY
 6     WORD DATA[2][3]={
 7       %0, %1, %2, %3, // [0][0],[0][1],[0][2],[0][3]
 8       %4, %5, %6, %7, // [1][0],[1][1],[1][2],[1][3]
 9       %8, %9,%10,%11  // [2][0],[2][1],[2][2],[2][3]
10     };
11
12   VAR I;
13
14   BEGIN
15     PRTARY2(DATA,2);
16   END;
17 //
18 //
19 PRTARY2(WORD ARY[][3],N)
20   VAR I,J;
21   BEGIN
22     FOR I=0 TO N {
23       FOR J=0 TO 3 {
24         PRINT("[",I,"][",J,"] : ");
25         PRINT(ARY[I][J]);
26         PRINT(/);
27       }
28     }
29   END;
30 //
31 //  END
32 //
```

リスト3　正誤例

```
//
//   私が掟だ!!.SL
//
MAIN()
  VAR A,B;

  BEGIN
    IF A{ BEEP(); }          (* 正しい *)
//  IF A[ BEEP(); ]          (* 間違い *)
//  IF A( BEEP(); )          (* 間違い *)

    IF A { BEEP(); }         (* 正しい *)
    IF A [ BEEP(); ]         (* 正しい *)
    IF A ( BEEP(); )         (* 正しい *)

    A=B/* コメント */+1;     (* 正しい *)
//  A=B(* コメント *)+1;     (* 間違い *)

    A=B /* コメント */+1;    (* 正しい *)
    A=B (* コメント *)+1;    (* 正しい *)
  END;
//
//  END
//
```

BYTE ARY[ ]

となっていますが，これが実引数として渡された配列のアドレスを受け取る部分で，1バイト型の間接変数ARYに配列のアドレスが代入されます。このように仮引数に間接変数を使えば，実引数として渡された配列を配列の形で直接アクセスできます。たとえば20行のように，

PRTARY(ARY1,3);

として関数PRTARYを呼び出すと，間接変数ARYには配列ARY1のアドレスが代入され40行のように

PRINT(ARY[I]);

などとすれば，ARY1[I]の内容を表示することができるわけです。しかもARY[I]というのはARY1[I]のコピーではなく実際にARY1[I]をアクセスしていますので，ARY[I]に数値を代入すればARY1[I]の内容も変更されます。このことを利用したのが，次のADDARYという関数で，実引数として渡された配列の各要素の値に3番目の引数として与えられた値を加えています。

間接変数を宣言する場合（仮引数も局所宣言の一種です）注意しなければならないのは，ふつうの変数の場合と違って変数名のみを書けばよいのではなく，型の指定と変数名の後ろに[ ]が必要だということです（間接変数だとコンパイラにわからせるため)。仮引数として間接変数を使用する場合は，型を実引数の配列の型に合わせる必要があります。実引数が1バイト型なら，仮引数も1バイト型にしなければなりません。

2次元配列も考え方はほぼ同じです。リスト2を見てください。リスト1同様最初に配列の宣言と初期化を行っています。初期化のしかたは1次元配列と同じなのですが，DATAという配列は2バイト型なので初期値の前に%がついています。%を取ると1バイト型の値となってしまいますので，2バイト型の配列を初期化する場合は気をつけてください（私もしょっちゅう%をつけ忘れる)。

PRTARY2は2次元配列を表示する関数ですが，実引数として2次元配列を渡す場合は仮引数も2次元の間接変数を使用します。2次元の間接変数は，

型　間接変数名［行］［列］

の形で宣言しますが，行の値は意味を持たないので，ふつう19行のように行を省略し，

WORD ARY[ ][3]

のように書きます。いい方を変えると「列の値は必ず書かなくてはいけない」ということです。このようにすれば25行のように，実引数として渡された2次元配列を2次元配列の形で直接アクセスすることができます。

この間接変数を発展させたものがC言語のポインタという考え方なのですが，SLANGではそこまで徹底していません。せいぜい「ポインタもどき」といったところです。間接変数はうまく使えばプログラムがコンパクトでわかりやすくなりますが，使いすぎると作った本人にさえ理解できない，それこそ神のみぞ知るといったプログラムができあがってしまいます。今回の例のような使い方が妥当なところでしょう。

さて，もうすでにSLANGでプログラムを組んでいる方も多いと思いますが，IF文などで間違っていないはずなのにIllegal name（名前の誤使用）のエラーが出たことはないでしょうか？　リスト3を見てください。文カッコに{を使用する場合は問題ないのですが，[や(の場合，直前が変数名などで間に空白などの区切りがないと，コンパイラはその変数を配列や関数と勘違いしてしまいエラーになってしまうのです。SLANGには配列名と[，関数名と(の間に空白を置いてはいけないという規則がありますが，逆にいえば，名前の直後に[や(を置けばその名前がたとえ変数名だったとしてもコンパイラは配列や関数として扱ってしまう，ということです。

このことにはコンパイラ制作中に気がついていたのですが，間接変数などとの関係で簡単に修正できず，またエラーも出るのであまり実害はないだろうと判断して，そういう仕様のままにしておきました。ところが私は通常，文カッコに{}を使用しますし，{の前に空白を置くクセがついていますので，その仕様自体をいつの間にか忘れてしまい，したがってリファレンスマニュアルからも当然のごとく抜けてしまったのでした。悩まれた方ゴメンナサイ。文カッコに[や(を使用する場合は直前に空白を置くか，条件式を( )でくくってください。

同様にこちらの場合はあまりないと思いますが，変数名などの直後に（*コメント*）を置く場合にも直前に空白が必要です。これらはハッキリいえばバグなのですが，作った本人がそういう仕様だといい切ってしまえば，もはやバグではないのだ。そう，

私が掟だ!! ウーム，なんて恐ろしい世界だろう。SLANGのプログラムは，どうか五月ちゃんやメイちゃんのように伸び伸びと，空白を入れて見通しよく書いてください。人が見てもわかりにくいようなプログラムは，コンパイラにもわかりにくいのです。

SLANGのテクニック

応用編として，2つばかりちょっとしたテクニックを紹介しましょう。マシン語の知識も必要ですので，初心者の方には難しいかもしれません。

リスト4は任意個数の引数を持つ関数の書き方の例です。SLANGでは引数の数をチェックしていますので，通常の方法ではC言語のprintfのような引数の個数が決まっていない関数を書くことができません。そこでMACHINE宣言で引数の数を省略したマシン語関数として宣言します。こうすれば引数の数のチェックは行われません。

問題はどうやって引数をアクセスするかですが，29行を見てください。このGETREGという関数がミソです。GETREGはその時点のレジスタの値を^Aや^HLなどの変数に代入する組み込み関数です。引数の数を省略して宣言したマシン語関数の場合，引数をスタックに積んで，引数の数はHLレジスタに代入されますから，変数^HLには引数の数が代入されることになります。

動的な局所変数や配列を使用する場合は関数の頭でIYレジスタを保存するためスタックにPUSHしますから，スタックの様子は図1のようになっています。変数^SPにはスタックポインタの値が代入されていますから，31行のように，

ARG＝^SP＋4＋^HL＊2；

とすれば，間接変数ARGには図1の①のアドレスが代入されます。こうすれば，n番目の引数は，

ARG[－n]

でアクセスすることができます。たとえば2番目の引数の値を表示したいのなら，

PRINT(ARG[－2])；

とすればよいのです。数値にマイナスがつくことと，ARGを必ず2バイト型の間接変数として宣言することを忘れないでください。

動的な局所変数や配列を使用しない場合はIYレジスタの退避は行われませんから，32行のように，

ARG＝^SP＋2＋^HL＊2；

となります。

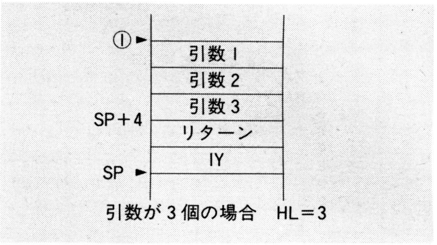
変身セット用のトランジェントコマンドをSLANGで書くことができます。リスト5はその例で，コマンドラインから与えられたパラメータをそのまま表示します。マシン語プログラムをトランジェントコマンドとするには，コマンドラインからパラメータを受け取ることと，エラーの場合Aレジスタにエラー番号をセットし，キャリフラグを立てて戻ることの2つの条件を満たさなければなりません。

コマンドラインからパラメータを受け取るのは非常に簡単です。トランジェントコマンドが呼び出されたとき，DEレジスタはパラメータの先頭のアドレスを指しています。SLANGでは実行時レジスタをまったくいじらずに関数MAINを呼び出しますから，MAINの先頭で22行のように，

GETREG( )；ARG＝^DE；

とすれば，1バイト型の間接変数ARGにパラメータのアドレスが代入されます。したがって，ARG[0]が1文字目，ARG[1]が2文字目，……となります。値が0なら，コマンドラインの終わりです。

図1 スタックの様子



リスト4 任意個の引数を使う

```
 1 //
 2 //   任意個数の引数を持つ関数
 3 //
 4 //         任意個数引数.SL
 5 //
 6   MACHINE SUB();
 7
 8 MAIN()
 9   BEGIN
10     PRINT("¥n¥nSUB(1,2,3);¥n¥n");
11
12     SUB(1,2,3);
13 //
14     PRINT("¥n¥nSUB(4,5,6,7,8);¥n¥n");
15
16     SUB(4,5,6,7,8);
17   END;
18 //
19 //
20 SUB()
21   VAR N, WORD ARG[];
22
23   VAR I;
24
25   BEGIN
26     VAR DUMMY;   (* 動的局所変数 *)
27 //
28 //
29     GETREG();
30     N=^HL;
31     ARG=^SP+4+^HL*2; (* ある場合 *)
32 //  ARG=^SP+2+^HL*2; (* ない場合 *)
33 //
34 //引数の数
35 //
36     PRINT("  ARG COUNT: ",N,"¥n¥n");
37 //
38     IF N==0 RETURN;
39 //
40 //引数の値
41 //
42     FOR I=1 TO N {
43       PRINT(
44         "  NO.",I," ARG : ",
45         ARG[-I],
46         /
47       );
48     }
49   END;
50 //
51 //  END
52 //
```

リスト5 コマンドの例

```
 1 //
 2 //   コマンドラインから引数を受けとる
 3 //
 4 //       CMD.SL
 5 //
 6 //       # CMD:ﾊﾟﾗﾒｰﾀ
 7 //
 8   CONST ABORTNO=3; (* 1 or 2 or 3 *)
 9
10 #IF ABORTNO==1
11
12   MACHINE ABORT(1);
13
14 #ENDIF
15
16 MAIN()
17   VAR BYTE ARG[];
18
19   VAR I;
20
21   BEGIN
22     GETREG();  ARG=^DE;
23 //
24     FOR(I=0; ARG[I]<>0; I++){
25
26       PRINT( CHR$( ARG[I] ) );
27     }
28 //
29     IF I==0 ABORT(14);
30 //
31     PRINT(/);
32   END;
33 //
34 //
35 //
36 #IF ABORTNO==1
37
38 ABORT( 1 )        (* HL:ERRNO *)
39   BEGIN
40     CODE($7D);  (* LD A,L   *)
41     STOP();
42   END;
43
44 #ENDIF
45 //
46 //
47 //
48 #IF ABORTNO==2
49
50 ABORT( ERRNO )
51   BEGIN
52     CODE(
53       [ERRNO],  (* HL:ERRNO *)
54       $7D       (* LD A,L   *)
55     );
56     STOP();
57   END;
58
59 #ENDIF
60 //
61 //
62 //
63 #IF ABORTNO==3
64
65 ABORT( ERRNO )
66   BEGIN
67     ^A=ERRNO;  CALL( &STOP() );
68   END;
69
70 #ENDIF
71 //
72 //  END
73 //
```

エラーのとき，キャリフラグを立てて戻るのも簡単で，単にSTOP( )で終了すればよいのです。SLANGでは，STOP( )で終了した場合のみキャリフラグが立つようになっています。

Aレジスタにエラー番号をセットする方法は，何通りか考えられます。ここでは，Aレジスタに値をセットしてSTOP( )で終了する関数ABORTを3種類ほど用意してみました。好きなものを使ってください。

1番目のABORTはもっとも省メモリタイプですが，プログラムの最初でMACHINE宣言をしなくてはなりません。STOP( )のように引数を持たない関数の呼び出しはCALL命令だけが生成されますから，直前でAレジスタに値を代入してやればいいわけです。

2番目のABORTはほぼ1番目のABORTと同じですが，こちらはMACHINE宣言の必要がありません。

3番目のABORTはもっともスマートなもので，CALL関数を使用しています。CALL関数はレジスタに値をセットしてマシン語のルーチンを呼び出すための組み込み関数ですから，変数^Aに値を代入すればAレジスタに値がセットすることになります。

わかりにくいと思われるのは，

&STOP( )

というところでしょう。&は演算子のひとつで，

&変数名

とすれば変数の格納されているアドレスを取り出しますが，SLANGでは関数名やラベル名にも使用することができます。関数名に使用する場合は引数を持つ持たないにかかわらず

&関数名( )

という形で使用します。&STOP( )はSTOP( )という関数のアドレスを指しますから，

CALL(&STOP( ))；

とすれば，STOP( )を呼び出せるわけです。もっともこれは，STOPが引数を持たない関数だから可能なのであって，引数を持つ関数の場合はこの方法は使えません。

最後に，ランタイムルーチンを拡張する方法を説明しましょう。ランタイムルーチンを拡張するのはマシン語のプログラムを組める方でないと無理ですし，そのくらいの力がある方ならばSLANGのソースリストを見れば，だいたいの見当がつくと思いますので，簡単に説明します。

まず，記号定数や外部のマシン語ルーチン（ほかのパッケージなど）を関数として登録する場合です。この場合はランタイムルーチンを追加する必要がありませんので非常に簡単です。66E0$_H$から始まる組込TBL1というテーブルに追加すればOKです。追加するデータは登録ずみのものを参考にしてください。

次はランタイムルーチンを追加する場合です。この場合は67AF$_H$からの組込TBL2にデータを追加します。マシン語ルーチンはランタイムルーチンの最後から続け，最終アドレスを300A$_H$に登録します。リロケートの際に書き換えてほしくないデータやワークエリアを含む場合は，66A1$_H$からの[DATA領域，DATA]に最初と最後のアドレスを追加登録してください。

さらにマシン語ルーチンを作るときに注意してほしいことがあります。ランタイムルーチンのリロケータは，ランタイムルーチン内のアドレスを2バイトのデータとして持つ命令のみ書き換えますから，アドレスを1バイトずつに分けてしまった場合は，書き換えの対象になりません。また未定義命令などもサポートしていません。

こういったことに気をつければ，自由にSLANGを拡張できます。

最後に

SLANGを駆け足で説明してきましたが，いかがでしたか。私自身あまりALGOL系の言語に慣れていませんので，少々わかりにくかったかもしれません（大きな声ではいえないが，私は一度もCやPASCALを使ったことがない）。

SLANGはCなどを指向して作った言語ですが，PRINT関数など意識してBASICに近づけた部分もあります。それはとっつきにくさを少しでも減らして，なるべく多くの方に使ってほしかったからです。そして整数しか使えないなどの欠点はありますが，実際に使ってもらえるだけのコンパイラに仕上げたつもりです。ゲームやユーティリティを作るのもよし，C言語の入門のつもりでもかまいません。使われてナンボのコンパイラ，どんどん使ってやってください。個人的には，思考型のゲームなどをSLANGで組んでくれるとうれしいなー。誰か，SLANGでオセロを作ってくれませんか？

セーブコマンドの修正

セーブコマンドにバグがありました。リスト6のようにプログラムを修正してください。

リスト6　セーブルーチン

```
0000                ;
0000                ; SLANG SAVE.Asm
0000                ;
0000                ; ]S filename:先頭:最終[:実行[:格納]]
0000                ;
0000                ;     []内は省略可
0000                ;
0000                      OFFSET $D000-$3148
0000
3148                      ORG $3148
3148                ;
3148                #HLHEX  EQU $1FB2
3148                #FILE   EQU $1FA3
3148                #DTADR  EQU $1F70
3148                #SIZE   EQU $1F72
3148                #EXADR  EQU $1F6E
3148                OFDATA  EQU $31B8
3148                ;
3148 3E 01                LD   A,1
314A CD A3 1F             CALL #FILE
314D                ;先頭
314D 1A FE 3A C0          LD   A,(DE)  IF A<>":" RET
3151 13                   INC  DE
3152 CD B2 1F D8          CALL #HLHEX  IF C RET
3156 22 70 1F             LD   (#DTADR),HL
3159 22 6E 1F             LD   (#EXADR),HL
315C 22 B8 31             LD   (OFDATA),HL
315F                ;最終
315F 1A FE 3A C0          LD   A,(DE)  IF A<>":" RET
3163 13                   INC  DE
3164 CD B2 1F D8          CALL #HLHEX  IF C RET
3168                ; OR A
3168 ED 4B 70 1F          LD   BC,(#DTADR)
316C ED 42                SBC  HL,BC
316E 23                   INC  HL
316F 22 72 1F             LD   (#SIZE),HL
3172                ;実行
3172 1A FE 3A 20 15       LD   A,(DE)  IF A<>":" JR LBL
3177 13                   INC  DE
3178 CD B2 1F D8          CALL #HLHEX  IF C RET
317C 22 6E 1F             LD   (#EXADR),HL
317F                ;格納
317F 1A FE 3A 20 08       LD   A,(DE)  IF A<>":" JR LBL
3184 13                   INC  DE
3185 CD B2 1F D8          CALL #HLHEX  IF C RET
3189 22 B8 31             LD   (OFDATA),HL
318C                LBL
318C                ;
318C                ; END
```

全機種共通S-OS“SWORD”要

マルチウィンドウドライバMW-1

S-OSでマルチウィンドウ処理を行うためのパッケージです。共通ルーチンで記述されていますので速度などではきつい面もありますが，このような処理の基本を探るという意味では参考になります。使い方次第ではS-OSの世界もまた広がりそうですね。

森　喜一郎 *Mori Kiichiro*

S-OSでウィンドウを

このプログラムはS-OSで「マルチウィンドウモドキ」を実現するためのものです。マルチウィンドウというと，すぐMacintoshの画面やX68000のビジュアルシェルのようなものを思い浮かべる方も多いでしょう。しかし，このパッケージではそのような完全なマルチウィンドウはできません。もともと，このパッケージでは以前PASOPIA7用“SWORD”のモニタに使われていたマルチウィンドウ「っぽい」環境を提供するためのものです。完全なマルチウィンドウといわないところがS-OSらしくていいんじゃないでしょうか。

入力と使用方法

各機種のモニタまたはMACINTO-Cなどのマシン語入力ツールからリスト1を打ち込んでください。チェックサム，CRCチェックバイトを確認のうえ，ディスクやテープにセーブしておきます。そして実行前には表3に従って“SWORD”内部を書き換えておきます。リスト1は単なるドライバですので，パラメータを与えることなく直接実行することは避けてください。使い方をよく理解しておかないとこれだけではなにもできません。

パッケージの使用方法ですが，まずDEレジスタにウィンドウの各種パラメータをセットしてB006Hをコールするとウィンドウがオープンされます。ウィンドウ内では#BOOT，#MON，#WIDCHを除くS-OSのすべてのサブルーチンを使用することができます。ウィンドウはメモリの許す限り，いくら開いてもかまいません。

逆にB003Hをコールすると新しいウィンドウから順番に閉じていきます。また，LレジスタにX座標，HレジスタにY座標を入れてB006Hをコールすることでその座標をウィンドウの右上とする位置へ現在使用中のウィンドウを移動することができます。

メモリ不足，不正パラメータなどでウィンドウが正常にオープンできなかった場合，キャリフラグを立ててリターンします。ウィンドウを開いたままリセットした場合などはワークエリアの内容が残っていますので異常動作することがあります。開いたウィンドウは必ず閉じるようにしたほうがよいでしょう。

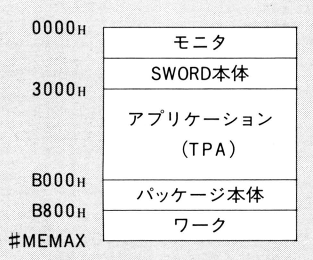
サンプル1はこのパッケージの使い方を示したものです。AF00Hをコールするとウィンドウがひとつ開かれます。次からはAF02Hをコールするたびにウィンドウの表示位置が変わっていきます。ウィンドウを消したいときはAF04Hをコールしてください。

プログラム解説

リスト2にパッケージ本体のソースリストを示します。まず最初（ウィンドウがまったくオープンされていない状態）にB000Hをコールすることで S-OSの表示関係の17個のルーチンにパッチがあてられます（とーとー，ここまでやってしまった)。これでS-OSの表示ルーチンはこのドライバに乗っとられたかたちとなり，すべてのウィンドウをクローズするまで，ウィンドウの内部で表示/入力が行われます。

ただし，各機種ごとにワークエリアのア

図1　メモリマップ





サンプル1

```
0000            ;*********************
0000            ;   WINDOW SAMPLE
0000            ;*********************
0000            WNOPEN  EQU  0B000H
0000            WCLOSE  EQU  0B003H
0000            WNMOVE  EQU  0B006H
0000            ;
AF00                    ORG  0AF00H
AF00            ;
AF00 18 04              JR   OPEN
AF02 18 09              JR   MOVE
AF04 18 23              JR   CLOSE
AF06            OPEN:
AF06 11 2D AF           LD   DE,DATA
AF09 CD 00 B0           CALL WNOPEN
AF0C C9                 RET
AF0D            MOVE:
AF0D 2A 3B AF           LD   HL,(ADR)
AF10 5E                 LD   E,(HL)
AF11 23                 INC  HL
AF12 56                 LD   D,(HL)
AF13 23                 INC  HL
AF14 22 3B AF           LD   (ADR),HL
AF17 EB                 EX   DE,HL
AF18 CD 06 B0           CALL WNMOVE
AF1B 21 3D AF           LD   HL,COUNT
AF1E 35                 DEC  (HL)
AF1F C0                 RET  NZ
AF20            ;
AF20 36 04              LD   (HL),4
AF22 21 3E AF           LD   HL,LDATA
AF25 22 3B AF           LD   (ADR),HL
AF28 C9                 RET
AF29            CLOSE:
AF29 CD 03 B0           CALL WCLOSE
AF2C C9                 RET
AF2D            ;
AF2D            DATA:
AF2D 20 43 5A           DEFM ' CZ-800C '
AF30 2D 38 30
AF33 30 43 20
AF36 00                 DEFB 0
AF37 05 05 14           DEFB 5,5,20,10
AF3A 0A
AF3B            ;
AF3B 3E AF      ADR:    DEFW LDATA
AF3D 04         COUNT:  DEFB 4
AF3E 14 0A      LDATA:  DEFB 20,10
AF40 0A 08              DEFB 10,8
AF42 18 00              DEFB 24,0
AF44 05 05              DEFB 5,5
```

```
AF00 18 04 18 09 18 23 11 2D : B6
AF08 AF CD 00 B0 C9 2A 3B AF : 09
AF10 5E 23 56 23 22 3B AF EB : F1
AF18 CD 06 B0 21 3D AF 35 C0 : 85
AF20 36 04 21 3E AF 22 3B AF : 54
AF28 C9 CD 03 B0 C9 20 43 5A : CF
AF30 2D 38 30 30 43 20 00 05 : 2D
AF38 05 14 0A 3E AF 04 14 0A : 32
AF40 0A 08 18 00 05 05       : 34
---------------------------------
SUM: 2D 1F 94 59 AF A2 C2 9F 0918
```

ドレスが異なる＃FPRNTだけは例外です。このルーチンは一律にパッチをあてることはできませんので“SWORD”内部を表3のように書き換えてください。

このパッケージは本当の意味でのマルチウィンドウではありません。ウィンドウプログラムに必須ともいうべき「仮想画面」を採用していないのです。これは画面退避用にS-OSの特殊ワークエリアを使用しているため，仮想画面を採用すると特殊ワークのもっとも小さいMZ-80Kなどの機種では少々難があると思ったためです。したがって，操作できるのはいちばん最近開いたウィンドウだけ，アクティブウィンドウを切り換えるといった高等な技は使えません。

ウィンドウ内では“SWORD”のコマンドはもちろん，少々手直しを加えれば（ウィンドウ内に収まるように)，いくつかのアプリケーションも走らせることができます。ただ，この場合には特殊ワークを使用するアプリケーション（ZEDAなど）では，B082$_H$からの2バイトがこのパッケージが使用する特殊ワークの先頭アドレスとなっていますので重複しないように書き換えてください。RAMディスクなどを使用する場合も同様です。

なお，ウィンドウの枠のデザインが気にいらない方は以下のキャラクタデータの格納アドレスを参考に書き換えるとよいでしょう。

左右上隅	B198$_H$
上枠	B197$_H$
左右枠	B1A0$_H$
左右下隅	B1AC$_H$
下枠	B1AB$_H$

表1　ジャンプテーブル一覧

アドレス（ラベル）	破壊	使用方法
B000$_H$ (WNOPEN)	AF	DEレジスタにパラメータの先頭アドレスをセットしてコールするとウィンドウが開く。
B003$_H$ (WCLOSE)		アクティブウィンドウを閉じる。
B006$_H$ (WNMOVE)		Lレジスタに X 座標，Hレジスタに Y 座標をセットしてコールすると，その位置にアクティブウィンドウが移動する。

表2　ウィンドウオープン時のパラメータ

```
DEFM   'ウィンドウ名'
DEFB    0
DEFB   左端X座標 (0≦X1≦79)
DEFB   上端Y座標 (0≦Y1≦24)
DEFB   右端X座標 (X1≦X2≦79)
DEFB   下端Y座標 (Y1≦Y2≦24)
```

表3　“SWORD”の変更点
それぞれのアドレスからの2バイトをF4$_H$, 1F$_H$に変更してください。

- X1/turbo
  1AC3$_H$, 1ACB$_H$, 1AD8$_H$
- MZ-80B/2000/2200/2500
  1849$_H$, 1851$_H$, 185E$_H$
- MZ-80K/C/1200/700/1500
  195F$_H$, 1967$_H$, 1974$_H$
- PC-8801(ROM)
  199E$_H$, 19A6$_H$, 19B3$_H$
- SMC-777
  064C$_H$, 0654$_H$, 0661$_H$
- MZ-2500
  1D8D$_H$, 1D95$_H$, 1DA2$_H$
- PC-8001/8801
  1766$_H$, 176E$_H$, 177B$_H$
- X1turbo
  0685$_H$, 068D$_H$, 069A$_H$
- PASOPIA7
  1B09$_H$, 1B11$_H$, 1B1E$_H$

最後に

このパッケージではポンポン画面を切り換えていくマルチウィンドウの醍醐味は味わえませんが，MZ-2500のアルゴ機能のようにアプリケーションの実行中にちょっとほかの処理がしたいといったときには威力を発揮します。

S-OS用サイドキックソフトなんてのが揃うとなかなか楽しくなりそうでしょう。機種によっては表示が重くなるのはしかたありませんが，できるだけスクロールなどをさせないような画面設計を心がけるなどすれば，それほどの問題はないと思われます。S-OSにもアルゴキーの代わりがほしいところですね。

Profile

◇森さんは大阪府にお住まいの22歳，大学4年生です。漢字出力パッケージでもお馴染みですね。大阪工大S-OSユーザーズグループの一員でもあります。

リスト1　MW-1ダンプリスト

```
B000 C3 DE B0 C3 16 B2 C3 98 : 37
B008 B2 00 00 00 00 00 00 00 : B2
B010 00 00 00 00 00 11 F5 1F : 25
B018 F2 1F EF 1F EC 1F E9 1F : 32
B020 E6 1F E3 1F E0 1F D4 1F : F9
B028 C2 1F BF 1F 19 20 1C 20 : 34
B030 1F 20 31 20 8F 1F 37 20 : 95
B038 00 00 00 00 00 00 00 00 : 00
B040 C3 26 B3 C3 96 B3 C3 9E : 09
B048 B3 C3 B6 B3 C3 C1 B3 C3 : D9
B050 CE B3 C3 DD B3 C3 EA B3 : 34
B058 C3 FC B3 C3 4B B4 C3 5E : 55
B060 B4 C3 67 B4 C3 7D B4 C3 : 49
B068 8F B4 C3 B1 B4 C3 B1 B4 : 93
B070 C3 B1 B4 00 00 00 00 00 : 28
B078 00 00 00 00 00 00 00 00 : 00
--------------------------------
SUM: 3B 1B 2F BB 58 6B 50 1E 7C95

B080 00 B8 00 00 00 00 00 00 : B8
B088 00 00 00 00 00 00 00 00 : 00
B090 00 00 00 00 00 00 00 00 : 00
B098 00 00 00 00 00 00 00 00 : 00
B0A0 00 00 00 00 00 00 00 00 : 00
B0A8 00 00 00 00 00 00 00 00 : 00
B0B0 00 00 00 00 00 00 00 00 : 00
B0B8 00 00 00 00 00 00 00 00 : 00
B0C0 00 00 00 00 00 00 00 00 : 00
B0C8 00 00 00 00 00 00 00 00 : 00
B0D0 00 00 00 00 00 00 00 00 : 00
B0D8 00 00 00 00 00 00 C5 D5 : 9A
B0E0 E5 DD E5 CD 33 B6 DD 2A : 64
B0E8 80 B0 06 0F 1A 13 B7 28 : 51
B0F0 0E DD 77 00 DD 23 10 F4 : 66
B0F8 1A 13 B7 20 FB 18 08 DD : FC
--------------------------------
SUM: 8D 35 19 FC 25 04 71 F8 9821
```

```
B100 36 00 00 DD 23 10 F8 DD : 1B
B108 2A 80 B0 06 04 1A 13 DD : 6E
B110 77 10 DD 23 10 F7 DD 2A : 95
B118 80 B0 CD B2 B4 DA 0F B2 : FE
B120 DD 70 14 DD 71 15 DD 46 : E7
B128 14 04 04 DD 4E 15 0C 0C : 74
B130 2A 82 B0 DD 75 1B DD 74 : 1A
B138 1C CD E4 B4 DA 0F B2 ED : 09
B140 53 82 B0 EB DD 6E 10 DD : A8
B148 66 11 CD F5 B4 21 D7 B0 : 95
B150 DD 7E 10 77 23 DD 7E 11 : 71
B158 77 23 DD 7E 14 77 23 DD : 80
B160 7E 15 77 23 7E 36 00 DD : BE
B168 77 16 23 7E 21 00 00 DD : 2C
B170 77 17 23 7E 36 00 DD 77 : B9
B178 18 CD 61 B0 DD 75 19 DD : 3E
--------------------------------
SUM: 1F 46 8E A7 73 DD ED D2 58CF

B180 74 1A DD 46 14 04 DD 4E : F4
B188 15 0C 0C DD 6E 10 DD 66 : CB
B190 11 CD 67 B0 11 86 B0 21 : 5D
B198 3D 2A CD 0C B6 0D 21 20 : 44
B1A0 3A CD 0C B6 0D 3E 01 B9 : CE
B1A8 20 F7 21 2D 2B CD 0C B6 : 1F
B1B0 2A 80 B0 06 00 D5 7E 23 : D6
B1B8 12 13 04 B7 20 F8 D1 DD : A6
B1C0 7E 14 3D B8 30 16 47 05 : 19
B1C8 DD 6E 10 2C 2C DD 66 11 : 07
B1D0 CD 67 B0 1A 13 CD 40 B0 : CE
B1D8 10 F9 18 16 C6 04 3D CB : 09
B1E0 3F 05 CB 38 DD 86 10 90 : 4A
B1E8 6F DD 66 11 CD 67 B0 CD : 74
B1F0 4F B0 01 1D 00 DD 09 DD : E0
B1F8 22 80 B0 3A 7F B0 3C 32 : 29
--------------------------------
SUM: C4 68 F5 33 FF BD 16 61 8CD2
```

```
B200 7F B0 21 00 00 CD 8F B4 : 60
B208 B7 DD E1 E1 D1 C1 C9 CD : 7E
B210 33 B6 AF 37 18 F3 3A 7F : 93
B218 B0 FE 01 D8 C5 D5 E5 DD : E3
B220 E5 DD 2A 80 B0 01 E3 FF : FF
B228 DD 09 DD 46 14 04 04 DD : 02
B230 4E 15 0C 0C DD 5E 1B DD : AE
B238 56 1C DD 6E 10 DD 66 11 : 21
B240 CD 10 B5 DD 6E 1B DD 66 : 3B
B248 1C 22 82 B0 DD 6E 19 DD : B1
B250 66 1A CD 67 B0 21 DB B0 : 10
B258 DD 7E 16 77 23 DD 7E 17 : 7D
B260 77 23 DD 7E 18 77 DD 22 : 83
B268 80 B0 3A 7F B0 3D 32 7F : 87
B270 B0 28 1B 01 E3 FF DD 09 : BC
B278 21 D7 B0 DD 7E 10 77 23 : AD
--------------------------------
SUM: 73 F4 9E 76 A6 E0 91 7E 2596

B280 DD 7E 11 77 23 DD 7E 14 : 75
B288 77 23 DD 7E 15 77 CD 33 : 81
B290 B6 DD E1 E1 D1 C1 B7 C9 : 67
B298 3A 7F B0 FE 01 D8 C5 D5 : DA
B2A0 E5 DD E5 DD 2A 80 B0 01 : DF
B2A8 E3 FF DD 09 DD 5E 12 DD : F2
B2B0 56 13 ED 53 84 B0 7D DD : 37
B2B8 77 10 DD 86 14 3C DD 77 : 8E
B2C0 12 7C DD 77 11 DD 86 15 : 6B
B2C8 3C DD 77 13 CD B2 B4 38 : 0E
B2D0 3E 04 04 0C 0C 2A 82 B0 : BA
B2D8 CD E4 B4 38 32 EB 2A D7 : BB
B2E0 B0 CD F5 B4 DD 5E 1B DD : 59
B2E8 56 1C CD 10 B5 DD 6E 10 : 5F
B2F0 DD 66 11 CD F5 B4 ED 5B : 12
B2F8 82 B0 CD 10 B5 EB CD 67 : E3
--------------------------------
SUM: 97 3C B7 02 01 35 0C 9A C946
```

```
B300 B4 ED 53 D7 B0 CD 8F B4 : 8B
B308 B7 DD E1 E1 D1 C1 C9 2A : DB
B310 D7 B0 DD 75 10 DD 74 11 : 4B
B318 ED 5B 84 B0 DD 73 12 DD : BB
B320 72 13 AF 37 18 E3 F5 E5 : 40
B328 FE 20 38 14 CD 40 B0 CD : F4
B330 67 B4 3A D9 B0 BD 20 5B : 16
B338 CD 85 B5 CD 9E B3 18 53 : 90
B340 FE 0D 20 05 CD 9E B3 18 : 66
B348 4A FE 0C 20 05 CD 47 B5 : 42
B350 18 41 CD 67 B4 FE 1C 20 : 7B
B358 0A 2C 3A D9 B0 BD CC 9E : 20
B360 B3 18 2D FE 1D 20 0D 2D : 6D
B368 7D FE FF 20 23 3A D9 B0 : 80
B370 3D 6F 18 04 FE 1E 20 09 : 0D
B378 25 7C FE FF 20 12 24 18 : 0C
---------------------------------
SUM: CF BA E0 54 35 21 C7 B5 F18A

B380 0F FE 1F 20 0E 24 3A DA : 92
B388 B0 BC 20 04 25 CD AF B5 : E6
B390 CD 8F B4 E1 F1 C9 F5 3E : DE
B398 20 CD 26 B3 F1 C9 F5 E5 : 5A
B3A0 CD 67 B4 24 3A DA B0 BC : 8C
B3A8 20 04 CD AF B5 25 2E 00 : A8
B3B0 CD 8F B4 E1 F1 C9 E5 CD : 5D
B3B8 67 B4 2C 2D C4 9E B3 E1 : 6A
B3C0 C9 F5 D5 1A 13 FE 0D 28 : F3
B3C8 11 CD 26 B3 18 F5 F5 D5 : 8E
B3D0 1A 13 B7 28 05 CD 26 B3 : B7
B3D8 18 F6 D1 F1 C9 E3 7E 23 : 1D
B3E0 B7 28 05 CD 26 B3 18 F6 : 98
B3E8 E3 C9 C5 E5 CD 67 B4 78 : B6
B3F0 95 47 3E 20 CD 26 B3 10 : F0
B3F8 FB E1 C1 C9 D5 E5 CD 21 : 0E
---------------------------------
SUM: 03 A8 C6 1A 47 B1 3B 8E 2CB8

B400 20 F5 CD CD 1F 28 3A F1 : 21
B408 FE 0D 28 05 CD 26 B3 18 : F6
B410 ED CD 67 B4 AF BC 28 07 : 6F
B418 25 CD 8F B5 20 F6 24 2E : 9E
B420 00 CD 7D B4 12 13 2C 3A : 89
B428 D9 B0 BD 20 F4 CD 8F B5 : 6B
B430 20 EC 1B 1A FE 20 28 FA : 81
B438 13 AF 12 CD 9E B3 E1 D1 : A4
B440 C9 CD 9E B3 F1 E1 D1 3E : C8
B448 1B 12 C9 F5 07 07 07 07 : 07
B450 CD BB 1F CD 26 B3 F1 CD : 0B
B458 BB 1F CD 26 B3 C9 7C CD : 92
B460 4B B4 7D CD 4B B4 C9 F5 : 06
B468 CD 61 B0 3A D7 B0 3C 95 : 70
B470 2F 3C 6F 3A D8 B0 3C 94 : 6C
B478 2F 3C 67 F1 C9 E5 3A D7 : 82
---------------------------------
SUM: 1E FA A8 C3 F1 10 BD CC 0867

B480 B0 3C 85 6F 3A D8 B0 3C : DE
B488 84 67 CD 64 B0 E1 C9 F5 : 6B
B490 E5 3A D9 B0 3D BD 38 16 : F0
B498 3A D7 B0 3C 85 6F 3A DA : 05
B4A0 B0 3D BC 38 09 3A D8 B0 : AC
B4A8 3C 84 67 CD 67 B0 E1 F1 : DD
B4B0 C9 C9 E5 DD 7E 12 21 5C : 61
B4B8 1F BE 30 25 DD 96 10 38 : ED
B4C0 20 28 1E 3D 47 FE 03 38 : 23
B4C8 18 DD 7E 13 21 5B 1F BE : DF
B4D0 30 0F DD 96 11 38 0A 28 : 2D
B4D8 08 3D 4F FE 02 38 02 E1 : AF
B4E0 C9 37 E1 C9 E5 59 50 CD : 05
B4E8 61 B6 19 ED 5B 68 1F EB : EA
B4F0 B7 ED 52 E1 C9 C5 D5 E5 : 1F
B4F8 E5 C5 CD 64 B0 EB CD 9A : DD
---------------------------------
SUM: 5D EC F4 A5 AB B1 14 8C 3ACD

B500 1F EB 13 2C 10 F4 C1 E1 : EF
B508 24 0D 20 EC E1 D1 C1 C9 : 79
B510 C5 D5 E5 CD 61 B0 22 84 : 03
B518 B0 E1 E5 E5 CD 67 B0 21 : 60
B520 86 B0 C5 EB CD 94 1F EB : 51
B528 13 77 23 10 F6 36 00 21 : 0A
B530 86 B0 EB CD 4F B0 EB C1 : 99
B538 E1 24 0D 20 DE 2A 84 B0 : 6E
B540 CD 67 B0 E1 D1 C1 C9 C5 : E5
B548 D5 E5 11 86 B0 3A D9 B0 : C4
B550 47 3E 20 12 13 10 FA AF : 83
B558 12 21 00 00 3A DA B0 47 : 3E
B560 11 86 B0 CD 8F B4 CD 4F : 73
B568 B0 24 10 F4 21 00 00 CD : C6
B570 8F B4 AF 21 DB B0 77 23 : 38
B578 77 23 77 21 00 00 CD 8F : 8E
---------------------------------
SUM: 7A D5 A4 2E 68 C9 3F 05 3705

B580 B4 E1 D1 C1 C9 F5 E5 CD : 97
B588 96 B5 B6 77 E1 F1 C9 E5 : F8
B590 CD 96 B5 A6 E1 C9 C5 E5 : 12
B598 24 3E 01 0F 25 20 FC E1 : 94
B5A0 CB 3C CB 3C CB 3C 06 00 : 1B
B5A8 4C 21 DB B0 09 C1 C9 C5 : 50
B5B0 D5 E5 3A D8 B0 67 3A D7 : F4
B5B8 B0 3C 6F 11 86 B0 3A DA : B6
B5C0 B0 3D 4F 28 1E 24 24 D5 : 9F
B5C8 E5 3A D9 B0 47 CD 64 B0 : D0
B5D0 2C 12 13 10 F8 AF 12 E1 : FB
B5D8 D1 25 CD 67 B0 CD 4F B0 : A6
B5E0 0D 20 E2 24 CD F6 B5 21 : CC
B5E8 DB B0 CB 26 23 CB 16 23 : A3
B5F0 CB 16 E1 D1 C1 C9 C5 E5 : C7
B5F8 3A D7 B0 3C 6F CD 67 B0 : 50
---------------------------------
SUM: 56 53 D2 68 E7 A7 92 DD 37E5

B600 3A D9 B0 47 CD 43 B0 10 : DA
B608 FB E1 C1 C9 C5 D5 E5 CD : B2
B610 61 B0 22 84 B0 E1 7C 12 : D6
B618 13 05 7D 12 13 10 FB 7C : 41
B620 12 13 AF 12 D1 CD 4F B0 : 83
B628 C1 E5 2A 84 B0 24 CD 67 : 5C
B630 B0 E1 C9 3A 7F B0 B7 C0 : 3A
B638 D5 DD 21 15 B0 DD 46 00 : BB
B640 DD 23 21 40 B0 23 DD 5E : 6F
B648 00 DD 56 01 1A 4E EB 71 : F8
B650 12 13 23 1A 4E EB 71 12 : 1E
B658 23 DD 23 DD 23 10 E6 D1 : EA
B660 C9 E5 21 00 00 7B 5A 16 : BA
B668 00 B7 28 0B CB 3F 30 01 : 25
B670 19 CB 23 CB 12 18 F2 5D : 4B
B678 54 E1 C9                : FE
---------------------------------
SUM: 49 5D C5 99 1D C5 C0 68 B56D
```

リスト2 MW-1ソースリスト

```
0000               ;********************************
0000               ;*  S-OS  Multi-window  system  *
0000               ;* (V1.1) 19,8/04/09  by K.MORI *
0000               ;********************************
0000               ;
0000               #PRNT    EQU   1FF4H
0000               #PRNTS   EQU   1FF1H
0000               #LTNL    EQU   1FEEH
0000               #NL      EQU   1FEBH
0000               #MSG     EQU   1FE8H
0000               #MSX     EQU   1FE5H
0000               #MPRNT   EQU   1FE2H
0000               #TAB     EQU   1FDFH
0000               #GETL    EQU   1FD3H
0000               #BRKEY   EQU   1FCDH
0000               #BELL    EQU   1FC4H
0000               #PRTHX   EQU   1FC1H
0000               #PRTHL   EQU   1FBEH
0000               #ASC     EQU   1FBBH
0000               #POKE    EQU   1F9AH
0000               #PEEK    EQU   1F94H
0000               #MON     EQU   1F8EH
0000               #CSR     EQU   2018H
0000               #SCRN    EQU   201BH
0000               #LOC     EQU   201EH
0000               #FLGET   EQU   2021H
0000               #WIDCH   EQU   2030H
0000               #BOOT    EQU   2036H
0000               ;
0000               #WKSIZ   EQU   1F68H
0000               #XYADR   EQU   1F78H
0000               #WIDTH   EQU   1F5CH
0000               #MXLIN   EQU   1F5BH
0000               ;
B000                        ORG   0B000H
B000               ;
B000               ;      ENTRY ADDRESS
B000               ;
B000               JENTRY:
B000 C3 DE B0               JP    WNOPEN
B003 C3 16 B2               JP    WCLOSE
B006 C3 98 B2               JP    WNMOVE
B009 00 00 00               DEFS 12                   ; RESERVED
B00C 00 00 00
B00F 00 00 00
B012 00 00 00
B015               ;
B015               ;      S-OS ADR.TABLE
B015               ;
B015               #TABLE:
B015 11                     DEFB 17                   ; KAZU
B016 F5 1F                  DEFW #PRNT+1
B018 F2 1F                  DEFW #PRNTS+1
B01A EF 1F                  DEFW #LTNL+1
B01C EC 1F                  DEFW #NL+1
B01E E9 1F                  DEFW #MSG+1
B020 E6 1F                  DEFW #MSX+1
B022 E3 1F                  DEFW #MPRNT+1
B024 E0 1F                  DEFW #TAB+1
B026 D4 1F                  DEFW #GETL+1
B028 C2 1F                  DEFW #PRTHX+1
B02A BF 1F                  DEFW #PRTHL+1
B02C 19 20                  DEFW #CSR+1
B02E 1C 20                  DEFW #SCRN+1
B030 1F 20                  DEFW #LOC+1
B032 31 20                  DEFW #WIDCH+1
B034 8F 1F                  DEFW #MON+1
B036 37 20                  DEFW #BOOT+1
B038 00 00 00               DEFS 8                    ; RESERVED
B03B 00 00 00
B03E 00 00
B040               ;
B040               ;      PATCH JUMP TABLE
B040               ;
B040               #PATCH:
B040 C3 26 B3      %PRNT:   JP    @PRNT
B043 C3 96 B3      %PRNTS:  JP    @PRNTS
B046 C3 9E B3      %LTNL:   JP    @LTNL
B049 C3 B6 B3      %NL:     JP    @NL
B04C C3 C1 B3      %MSG:    JP    @MSG
B04F C3 CE B3      %MSX:    JP    @MSX
B052 C3 DD B3      %MPRNT:  JP    @MPRNT
B055 C3 EA B3      %TAB:    JP    @TAB
B058 C3 FC B3      %GETL:   JP    @GETL
B05B C3 4B B4      %PRTHX:  JP    @PRTHX
B05E C3 5E B4      %PRTHL:  JP    @PRTHL
B061 C3 67 B4      %CSR:    JP    @CSR
B064 C3 7D B4      %SCRN:   JP    @SCRN
B067 C3 8F B4      %LOC:    JP    @LOC
B06A C3 B1 B4      %WIDCH:  JP    @WIDCH
B06D C3 B1 B4      %MON:    JP    @MON
B070 C3 B1 B4      %BOOT:   JP    @BOOT
B073 00 00 00               DEFS 12                   ; RESERVED
B076 00 00 00
B079 00 00 00
B07C 00 00 00
B07F               ;
B07F               ;      WORK AREA
B07F               ;
B07F 00            WINFLG: DEFB 0                     ; number of windows
B080 00 B8         PARPTR: DEFW WIB                   ; top of parameter area
B082 00 00         WKADR:  DEFW 0                     ; top of work area
B084 00 00         WORDBF: DEFS 2                     ; etc. work
B086 00 00 00      STRBUF: DEFS 81                    ; line work
B089 00 00 00
B08C 00 00 00
B08F 00 00 00
B092 00 00 00
B095 00 00 00
B098 00 00 00
B09B 00 00 00
B09E 00 00 00
B0A1 00 00 00
B0A4 00 00 00
B0A7 00 00 00
B0AA 00 00 00
B0AD 00 00 00
B0B0 00 00 00
B0B3 00 00 00
B0B6 00 00 00
B0B9 00 00 00
B0BC 00 00 00
B0BF 00 00 00
B0C2 00 00 00
B0C5 00 00 00
B0C8 00 00 00
B0CB 00 00 00
B0CE 00 00 00
B0D1 00 00 00
B0D4 00 00 00
B0D7               ;
B0D7 00 00         @WINXY: DEFS 2                     ; (X1,Y1) of window
B0D9 00            @WIDTH: DEFS 1                     ; width of window
B0DA 00            @MXLIN: DEFS 1                     ; max line no.
B0DB 00 00 00      @LCFLG: DEFS 3                     ; line connect flag
B0DE               ;
B0DE               ;******************************
B0DE               ;*      OPEN THE WINDOW        *
```


▶ハードの進歩に伴ってだんだんパソコンのつくりが複雑になってきている。MZ-80が自分の手足のようになっていたのに，いまでは私がXlturboの手足である。これから数年のち，パソコンのハードは市販のマシン語ゲームのようになってしまうのだろうか。

矢落　亮一（16）香川県

```
B0DE                ;*****************************
B0DE                ;
B0DE                WNOPEN:
B0DE C5                     PUSH BC
B0DF D5                     PUSH DE
B0E0 E5                     PUSH HL
B0E1 DD E5                  PUSH IX
B0E3 CD 33 B6               CALL PATCH
B0E6                ;
B0E6                ;       SET WINDOW NAME
B0E6                ;
B0E6 DD 2A 80               LD   IX,(PARPTR)       ; IX+0 = WNAME
B0E9 B0
B0EA 06 0F                  LD   B,15
B0EC                SETL1:
B0EC 1A                     LD   A,(DE)
B0ED 13                     INC  DE
B0EE B7                     OR   A
B0EF 28 0E                  JR   Z,SETL3
B0F1 DD 77 00               LD   (IX+0),A
B0F4 DD 23                  INC  IX
B0F6 10 F4                  DJNZ SETL1
B0F8                SETL2:
B0F8 1A                     LD   A,(DE)
B0F9 13                     INC  DE
B0FA B7                     OR   A
B0FB 20 FB                  JR   NZ,SETL2
B0FD 18 08                  JR   SETXY
B0FF                SETL3:
B0FF DD 36 00               LD   (IX+0),0
B102 00
B103 DD 23                  INC  IX
B105 10 F8                  DJNZ SETL3
B107                ;
B107                ;       SET WINDOW POSITION
B107                ;
B107                SETXY:
B107 DD 2A 80               LD   IX,(PARPTR)
B10A B0
B10B 06 04                  LD   B,4
B10D                SETL4:
B10D 1A                     LD   A,(DE)
B10E 13                     INC  DE
B10F DD 77 10               LD   (IX+16),A         ; IX+16 = WINXY
B112 DD 23                  INC  IX
B114 10 F7                  DJNZ SETL4
B116                ;
B116 DD 2A 80               LD   IX,(PARPTR)
B119 B0
B11A                ;
B11A                ;       SET WIDTH,MAXLIN
B11A                ;
B11A CD B2 B4               CALL WIDLIN
B11D DA 0F B2               JP   C,OPERR
B120 DD 70 14               LD   (IX+20),B         ; IX+20 = WIDTH
B123 DD 71 15               LD   (IX+21),C         ; IX+21 = MXLIN
B126                ;
B126                ;       SCREEN ==> WORK
B126                ;
B126 DD 46 14               LD   B,(IX+20)         ; IX+20 = WIDTH
B129 04                     INC  B
B12A 04                     INC  B
B12B DD 4E 15               LD   C,(IX+21)         ; IX+21 = MXLIN
B12E 0C                     INC  C
B12F 0C                     INC  C
B130 2A 82 B0               LD   HL,(WKADR)        ; HL <== work ptr.
B133 DD 75 1B               LD   (IX+27),L         ; IX+27 = WKPTR
B136 DD 74 1C               LD   (IX+28),H
B139 CD E4 B4               CALL WKCHK             ; DE <== next work ptr.
B13C DA 0F B2               JP   C,OPERR
B13F                ;
B13F ED 53 82               LD   (WKADR),DE        ; store work ptr.
B142 B0
B143 EB                     EX   DE,HL             ; DE <== work ptr.
B144 DD 6E 10               LD   L,(IX+16)         ; L <== X
B147 DD 66 11               LD   H,(IX+17)         ; H <== Y
B14A CD F5 B4               CALL STSCRN
B14D                ;
B14D                ;       GET PARAMETER
B14D                ;
B14D 21 D7 B0               LD   HL,@WINXY
B150 DD 7E 10               LD   A,(IX+16)         ; IX+16 = WINXY
B153 77                     LD   (HL),A            ; HL   = @WINXY
B154 23                     INC  HL
B155 DD 7E 11               LD   A,(IX+17)
B158 77                     LD   (HL),A
B159 23                     INC  HL
B15A                ;
B15A DD 7E 14               LD   A,(IX+20)         ; IX+20 = WIDTH
B15D 77                     LD   (HL),A            ; HL   = @WIDTH
B15E 23                     INC  HL
B15F DD 7E 15               LD   A,(IX+21)         ; IX+21 = MXLIN
B162 77                     LD   (HL),A            ; HL   = @MXLIN
B163 23                     INC  HL
B164                ;
B164 7E                     LD   A,(HL)            ; HL   = @LCFLG
B165 36 00                  LD   (HL),0
B167 DD 77 16               LD   (IX+22),A         ; IX+22 = LCBUF
B16A 23                     INC  HL
B16B 7E                     LD   A,(HL)
B16C 21 00 00               LD   HL,0
B16F DD 77 17               LD   (IX+23),A
B172 23                     INC  HL
B173 7E                     LD   A,(HL)
B174 36 00                  LD   (HL),0
B176 DD 77 18               LD   (IX+24),A
B179                ;
B179 CD 61 B0               CALL %CSR
B17C DD 75 19               LD   (IX+25),L         ; IX+25 = XYBUF
B17F DD 74 1A               LD   (IX+26),H
B182                ;
B182                ;       DISP. THE WINDOW
B182                ;
B182 DD 46 14               LD   B,(IX+20)         ; IX+20 = WIDTH
B185 04                     INC  B
B186 DD 4E 15               LD   C,(IX+21)         ; IX+21 = MXLIN
B189 0C                     INC  C
B18A 0C                     INC  C
B18B                ;
B18B DD 6E 10               LD   L,(IX+16)         ; IX+16 = WINXY
B18E DD 66 11               LD   H,(IX+17)
B191 CD 67 B0               CALL %LOC              ; CURSOR L,H
B194 11 86 B0               LD   DE,STRBUF
B197 21 3D 2A               LD   HL,'*='
B19A CD 0C B6               CALL LINPRT
B19D 0D                     DEC  C
B19E 21 20 3A               LD   HL,': '
B1A1                DSL1:
B1A1 CD 0C B6               CALL LINPRT
B1A4 0D                     DEC  C
B1A5 3E 01                  LD   A,1
B1A7 B9                     CP   C
B1A8 20 F7                  JR   NZ,DSL1
B1AA 21 2D 2B               LD   HL,'+-'
B1AD CD 0C B6               CALL LINPRT
B1B0                ;
B1B0 2A 80 B0               LD   HL,(PARPTR)       ; (PARPTR) = WNAME
B1B3 06 00                  LD   B,0
B1B5 D5                     PUSH DE
B1B6                DSL2:
B1B6 7E                     LD   A,(HL)
B1B7 23                     INC  HL
B1B8 12                     LD   (DE),A
B1B9 13                     INC  DE
B1BA 04                     INC  B                 ; B <== LEN.
B1BB B7                     OR   A
B1BC 20 F8                  JR   NZ,DSL2           ; until A=0
B1BE                ;
B1BE D1                     POP  DE
B1BF DD 7E 14               LD   A,(IX+20)         ; IX+20 = WIDTH
B1C2 3D                     DEC  A
B1C3 B8                     CP   B                 ; WID : LEN
B1C4 30 16                  JR   NC,DSL5
B1C6                ;                              ; case W < L
B1C6 47                     LD   B,A
B1C7 05                     DEC  B
B1C8 DD 6E 10               LD   L,(IX+16)         ; IX+16 = WINXY
B1CB 2C                     INC  L
B1CC 2C                     INC  L
B1CD DD 66 11               LD   H,(IX+17)
B1D0 CD 67 B0               CALL %LOC
B1D3                DSL4:
B1D3 1A                     LD   A,(DE)
B1D4 13                     INC  DE
B1D5 CD 40 B0               CALL %PRNT
B1D8 10 F9                  DJNZ DSL4
B1DA 18 16                  JR   DSL6
B1DC                DSL5:                          ; case W >= L
B1DC C6 04                  ADD  A,4
B1DE 3D                     DEC  A
B1DF CB 3F                  SRL  A
B1E1 05                     DEC  B
B1E2 CB 38                  SRL  B
B1E4 DD 86 10               ADD  A,(IX+16)         ; IX+16 = WINXY
B1E7 90                     SUB  B
B1E8 6F                     LD   L,A               ; L <== X
B1E9 DD 66 11               LD   H,(IX+17)         ; H <== Y
B1EC CD 67 B0               CALL %LOC
B1EF CD 4F B0               CALL %MSX
B1F2                DSL6:
B1F2 01 1D 00               LD   BC,29
B1F5 DD 09                  ADD  IX,BC
B1F7 DD 22 80               LD   (PARPTR),IX
B1FA B0
B1FB 3A 7F B0               LD   A,(WINFLG)
B1FE 3C                     INC  A
B1FF 32 7F B0               LD   (WINFLG),A
B202                ;
B202 21 00 00               LD   HL,0
B205 CD 8F B4               CALL @LOC
B208 B7                     OR   A
B209                OPE:
B209 DD E1                  POP  IX
B20B E1                     POP  HL
B20C D1                     POP  DE
B20D C1                     POP  BC
B20E C9                     RET
B20F                OPERR:
B20F CD 33 B6               CALL PATCH
B212 AF                     XOR  A
B213 37                     SCF
B214 18 F3                  JR   OPE
B216                ;
B216                ;**************************
B216                ;*      WINDOW CLOSE      *
B216                ;**************************
B216                ;
B216                WCLOSE:
B216 3A 7F B0               LD   A,(WINFLG)
B219 FE 01                  CP   1
B21B D8                     RET  C
B21C                ;
B21C C5                     PUSH BC
B21D D5                     PUSH DE
B21E E5                     PUSH HL
B21F DD E5                  PUSH IX
B221                ;
B221 DD 2A 80               LD   IX,(PARPTR)
B224 B0
B225 01 E3 FF               LD   BC,-29
B228 DD 09                  ADD  IX,BC
B22A                ;
B22A                ;       WORK ==> SCREEN
B22A                ;
B22A DD 46 14               LD   B,(IX+20)         ; IX+20 = WIDTH
B22D 04                     INC  B
B22E 04                     INC  B
B22F DD 4E 15               LD   C,(IX+21)         ; IX+21 = MXLIN
B232 0C                     INC  C
B233 0C                     INC  C
B234 DD 5E 1B               LD   E,(IX+27)         ; IX+27 = WKPTR
B237 DD 56 1C               LD   D,(IX+28)
B23A DD 6E 10               LD   L,(IX+16)         ; IX+16 = WINXY
B23D DD 66 11               LD   H,(IX+17)
B240 CD 10 B5               CALL LDSCRN
B243                ;
B243 DD 6E 1B               LD   L,(IX+27)         ; IX+27 = WKPTR
B246 DD 66 1C               LD   H,(IX+28)
B249 22 82 B0               LD   (WKADR),HL
B24C DD 6E 19               LD   L,(IX+25)         ; IX+25 = XYBUF
B24F DD 66 1A               LD   H,(IX+26)
B252 CD 67 B0               CALL %LOC
B255 21 DB B0               LD   HL,@LCFLG
B258 DD 7E 16               LD   A,(IX+22)         ; IX+22 = LCBUF
B25B 77                     LD   (HL),A
B25C 23                     INC  HL
B25D DD 7E 17               LD   A,(IX+23)
B260 77                     LD   (HL),A
B261 23                     INC  HL
B262 DD 7E 18               LD   A,(IX+24)
B265 77                     LD   (HL),A
B266 DD 22 80               LD   (PARPTR),IX
B269 B0
B26A                ;
B26A 3A 7F B0               LD   A,(WINFLG)
B26D 3D                     DEC  A
B26E 32 7F B0               LD   (WINFLG),A
B271 28 1B                  JR   Z,CLE
B273                ;
B273                ;       GET PARAMETER
B273                ;
B273 01 E3 FF               LD   BC,-29
B276 DD 09                  ADD  IX,BC
B278 21 D7 B0               LD   HL,@WINXY
B27B DD 7E 10               LD   A,(IX+16)         ; IX+16 = WINXY
B27E 77                     LD   (HL),A
B27F 23                     INC  HL
B280 DD 7E 11               LD   A,(IX+17)
B283 77                     LD   (HL),A
B284 23                     INC  HL
B285 DD 7E 14               LD   A,(IX+20)         ; IX+20 = WIDTH
B288 77                     LD   (HL),A
B289 23                     INC  HL
B28A DD 7E 15               LD   A,(IX+21)         ; IX+21 = MXLIN
B28D 77                     LD   (HL),A
B28E                CLE:
```

▶『歴史街道』という雑誌の６月号の表紙にはMZ-731としか思えない（多少おかしなところはありますが）マシンの絵が出ています。やはりMZ-700は歴史の一部だったんですね。

綾塚　祐二（18）東京都

```
B28E CD 33 B6           CALL PATCH
B291 DD E1              POP  IX
B293 E1                 POP  HL
B294 D1                 POP  DE
B295 C1                 POP  BC
B296 B7                 OR   A
B297 C9                 RET
B298               ;
B298               ;******************************
B298               ;*  MOVE WINDOW ( HL = YX )   *
B298               ;******************************
B298               ;
B298               WNMOVE:
B298 3A 7F B0           LD   A,(WINFLG)
B29B FE 01              CP   1
B29D D8                 RET  C
B29E               ;
B29E C5                 PUSH BC
B29F D5                 PUSH DE
B2A0 E5                 PUSH HL
B2A1 DD E5              PUSH IX
B2A3               ;
B2A3 DD 2A 80           LD   IX,(PARPTR)
B2A6 B0
B2A7 01 E3 FF           LD   BC,-29
B2AA DD 09              ADD  IX,BC
B2AC DD 5E 12           LD   E,(IX+18)
B2AF DD 56 13           LD   D,(IX+19)
B2B2 ED 53 84           LD   (WORDBF),DE
B2B5 B0
B2B6               ;
B2B6 7D                 LD   A,L
B2B7 DD 77 10           LD   (IX+16),A         ; IX+16 = WINXY
B2BA DD 86 14           ADD  A,(IX+20)         ; IX+20 = WIDTH
B2BD 3C                 INC  A
B2BE DD 77 12           LD   (IX+18),A
B2C1 7C                 LD   A,H
B2C2 DD 77 11           LD   (IX+17),A
B2C5 DD 86 15           ADD  A,(IX+21)         ; IX+21 = MXLIN
B2C8 3C                 INC  A
B2C9 DD 77 13           LD   (IX+19),A
B2CC               ;
B2CC CD B2 B4           CALL WIDLIN            ; BC <== WID,LIN
B2CF 38 3E              JR   C,MVERR
B2D1 04                 INC  B
B2D2 04                 INC  B
B2D3 0C                 INC  C
B2D4 0C                 INC  C
B2D5 2A 82 B0           LD   HL,(WKADR)
B2D8 CD E4 B4           CALL WKCHK
B2DB 38 32              JR   C,MVERR
B2DD               ;
B2DD EB                 EX   DE,HL             ; DE <== (WKADR)
B2DE 2A D7 B0           LD   HL,(@WINXY)
B2E1 CD F5 B4           CALL STSCRN
B2E4               ;
B2E4 DD 5E 1B           LD   E,(IX+27)         ; IX+27 = WKPTR
B2E7 DD 56 1C           LD   D,(IX+28)
B2EA CD 10 B5           CALL LDSCRN
B2ED               ;
B2ED DD 6E 10           LD   L,(IX+16)
B2F0 DD 66 11           LD   H,(IX+17)
B2F3 CD F5 B4           CALL STSCRN
B2F6               ;
B2F6 ED 5B 82           LD   DE,(WKADR)
B2F9 B0
B2FA CD 10 B5           CALL LDSCRN
B2FD EB                 EX   DE,HL
B2FE CD 67 B4           CALL @CSR
B301 ED 53 D7           LD   (@WINXY),DE
B304 B0
B305 CD 8F B4           CALL @LOC
B308 B7                 OR   A
B309               MVE:
B309 DD E1              POP  IX
B30B E1                 POP  HL
B30C D1                 POP  DE
B30D C1                 POP  BC
B30E C9                 RET
B30F               MVERR:
B30F 2A D7 B0           LD   HL,(@WINXY)
B312 DD 75 10           LD   (IX+16),L
B315 DD 74 11           LD   (IX+17),H
B318 ED 5B 84           LD   DE,(WORDBF)
B31B B0
B31C DD 73 12           LD   (IX+18),E
B31F DD 72 13           LD   (IX+19),D
B322 AF                 XOR  A
B323 37                 SCF
B324 18 E3              JR   MVE
B326               ;
B326               ;********************************
B326               ;*      PATCH SUBROUTINES       *
B326               ;********************************
B326               ;
B326               @PRNT:
B326 F5                 PUSH AF
B327 E5                 PUSH HL
B328 FE 20              CP   ' '
B32A 38 14              JR   C,@PRT1
B32C               ;
B32C CD 40 B0           CALL %PRNT             ; Acc print
B32F CD 67 B4           CALL @CSR              ; HL <== YX
B332 3A D9 B0           LD   A,(@WIDTH)
B335 BD                 CP   L
B336 20 5B              JR   NZ,@PRTE
B338 CD 85 B5           CALL LNCNT
B33B CD 9E B3           CALL @LTNL
B33E 18 53              JR   @PRTE
B340               @PRT1:
B340 FE 0D              CP   13
B342 20 05              JR   NZ,@PRT2
B344 CD 9E B3           CALL @LTNL
B347 18 4A              JR   @PRTE
B349               @PRT2:
B349 FE 0C              CP   12
B34B 20 05              JR   NZ,@PRT3
B34D CD 47 B5           CALL CLS
B350 18 41              JR   @PRTE
B352               @PRT3:
B352 CD 67 B4           CALL @CSR              ; HL <== YX
B355 FE 1C              CP   28
B357 20 0A              JR   NZ,@PRT4
B359 2C                 INC  L
B35A 3A D9 B0           LD   A,(@WIDTH)
B35D BD                 CP   L
B35E CC 9E B3           CALL Z,@LTNL
B361 18 2D              JR   @PRT8
B363               @PRT4:
B363 FE 1D              CP   29
B365 20 0D              JR   NZ,@PRT5
B367 2D                 DEC  L
B368 7D                 LD   A,L
B369 FE FF              CP   0FFH
B36B 20 23              JR   NZ,@PRT8
B36D 3A D9 B0           LD   A,(@WIDTH)
B370 3D                 DEC  A
B371 6F                 LD   L,A
B372 18 04              JR   @PRT6
B374               @PRT5:
B374 FE 1E              CP   30
B376 20 09              JR   NZ,@PRT7
B378               @PRT6:
B378 25                 DEC  H
B379 7C                 LD   A,H
B37A FE FF              CP   0FFH
B37C 20 12              JR   NZ,@PRT8
B37E 24                 INC  H
B37F 18 0F              JR   @PRT8
B381               @PRT7:
B381 FE 1F              CP   31
B383 20 0E              JR   NZ,@PRTE
B385 24                 INC  H
B386 3A DA B0           LD   A,(@MXLIN)
B389 BC                 CP   H
B38A 20 04              JR   NZ,@PRT8
B38C 25                 DEC  H
B38D CD AF B5           CALL SCROLL
B390               @PRT8:
B390 CD 8F B4           CALL @LOC              ; HL ==> YX
B393               @PRTE:
B393 E1                 POP  HL
B394 F1                 POP  AF
B395 C9                 RET
B396               ;
B396               @PRNTS:
B396 F5                 PUSH AF
B397 3E 20              LD   A,' '
B399 CD 26 B3           CALL @PRNT
B39C F1                 POP  AF
B39D C9                 RET
B39E               ;
B39E               @LTNL:
B39E F5                 PUSH AF
B39F E5                 PUSH HL
B3A0 CD 67 B4           CALL @CSR              ; HL <== YX
B3A3 24                 INC  H
B3A4 3A DA B0           LD   A,(@MXLIN)
B3A7 BC                 CP   H
B3A8 20 04              JR   NZ,@NL1           ; not over bottom
B3AA CD AF B5           CALL SCROLL
B3AD 25                 DEC  H
B3AE               @NL1:
B3AE 2E 00              LD   L,0
B3B0 CD 8F B4           CALL @LOC              ; HL ==> YX
B3B3 E1                 POP  HL
B3B4 F1                 POP  AF
B3B5 C9                 RET
B3B6               ;
B3B6               @NL:
B3B6 E5                 PUSH HL
B3B7 CD 67 B4           CALL @CSR
B3BA 2C                 INC  L
B3BB 2D                 DEC  L
B3BC C4 9E B3           CALL NZ,@LTNL
B3BF E1                 POP  HL
B3C0 C9                 RET
B3C1               ;
B3C1               @MSG:
B3C1 F5                 PUSH AF
B3C2 D5                 PUSH DE
B3C3               @MSGL:
B3C3 1A                 LD   A,(DE)
B3C4 13                 INC  DE
B3C5 FE 0D              CP   13
B3C7 28 11              JR   Z,@MSE
B3C9 CD 26 B3           CALL @PRNT
B3CC 18 F5              JR   @MSGL
B3CE               ;
B3CE F5            @MSX:  PUSH AF
B3CF D5                 PUSH DE
B3D0               @MSXL:
B3D0 1A                 LD   A,(DE)
B3D1 13                 INC  DE
B3D2 B7                 OR   A
B3D3 28 05              JR   Z,@MSE
B3D5 CD 26 B3           CALL @PRNT
B3D8 18 F6              JR   @MSXL
B3DA               @MSE:
B3DA D1                 POP  DE
B3DB F1                 POP  AF
B3DC C9                 RET
B3DD               ;
B3DD               @MPRNT:
B3DD E3                 EX   (SP),HL
B3DE               @MPRL:
B3DE 7E                 LD   A,(HL)
B3DF 23                 INC  HL
B3E0 B7                 OR   A
B3E1 28 05              JR   Z,@MPRE
B3E3 CD 26 B3           CALL @PRNT
B3E6 18 F6              JR   @MPRL
B3E8               @MPRE:
B3E8 E3                 EX   (SP),HL
B3E9 C9                 RET
B3EA               ;
B3EA               @TAB:
B3EA C5                 PUSH BC
B3EB E5                 PUSH HL
B3EC CD 67 B4           CALL @CSR
B3EF 78                 LD   A,B
B3F0 95                 SUB  L
B3F1 47                 LD   B,A
B3F2 3E 20              LD   A,' '
B3F4               @TABL:
B3F4 CD 26 B3           CALL @PRNT
B3F7 10 FB              DJNZ @TABL
B3F9 E1                 POP  HL
B3FA C1                 POP  BC
B3FB C9                 RET
B3FC               ;
B3FC               @GETL:
B3FC D5                 PUSH DE
B3FD E5                 PUSH HL
B3FE               @GETLL:
B3FE CD 21 20           CALL #FLGET
B401 F5                 PUSH AF
B402 CD CD 1F           CALL #BRKEY
B405 28 3A              JR   Z,@GETB
B407 F1                 POP  AF
B408 FE 0D              CP   13
B40A 28 05              JR   Z,@GETL1
B40C CD 26 B3           CALL @PRNT
B40F 18 ED              JR   @GETLL
B411               @GETL1:
B411 CD 67 B4           CALL @CSR              ; HL <== YX
B414               @GETL2:
B414 AF                 XOR  A
B415 BC                 CP   H
B416 28 07              JR   Z,@GETL4
B418 25                 DEC  H                 ; Y <== Y - 1
B419 CD 8F B5           CALL ?LCNT             ; line connect?
B41C 20 F6              JR   NZ,@GETL2
B41E               @GETL3:
B41E 24                 INC  H
B41F               @GETL4:
```

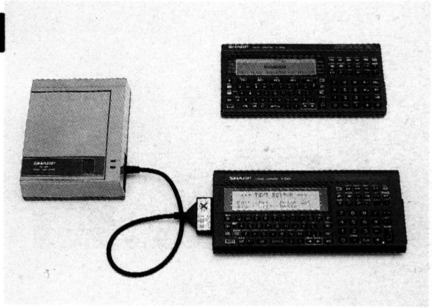
▶するってぇとα-アジールのサイコフレームはV8の後継機種に開発されたとおっしゃるんで。それにしても88がクツの裏なんて，それじゃーあんまりクツの裏がかわいそうすぎます。
龍尾　謙二（21）岐阜県

```
B41F 2E 00                 LD   L,0
B421             @GETL5:
B421 CD 7D B4              CALL @SCRN            ; A <== (HL)
B424 12                    LD   (DE),A
B425 13                    INC  DE
B426 2C                    INC  L
B427 3A D9 B0              LD   A,(@WIDTH)
B42A BD                    CP   L
B42B 20 F4                 JR   NZ,@GETL5
B42D CD 8F B5              CALL ?LCNT
B430 20 EC                 JR   NZ,@GETL3
B432             @GETL6:
B432 1B                    DEC  DE
B433 1A                    LD   A,(DE)
B434 FE 20                 CP   ' '
B436 28 FA                 JR   Z,@GETL6
B438 13                    INC  DE
B439 AF                    XOR  A
B43A 12                    LD   (DE),A
B43B CD 9E B3              CALL @LTNL
B43E E1                    POP  HL
B43F D1                    POP  DE
B440 C9                    RET
B441             @GETB:
B441 CD 9E B3              CALL @LTNL
B444 F1                    POP  AF
B445 E1                    POP  HL
B446 D1                    POP  DE
B447 3E 1B                 LD   A,1BH
B449 12                    LD   (DE),A
B44A C9                    RET
B44B             ;
B44B             @PRTHX:
B44B F5                    PUSH AF
B44C 07                    RLCA
B44D 07                    RLCA
B44E 07                    RLCA
B44F 07                    RLCA
B450 CD BB 1F              CALL #ASC
B453 CD 26 B3              CALL @PRNT
B456 F1                    POP  AF
B457 CD BB 1F              CALL #ASC
B45A CD 26 B3              CALL @PRNT
B45D C9                    RET
B45E             ;
B45E             @PRTHL:
B45E 7C                    LD   A,H
B45F CD 4B B4              CALL @PRTHX
B462 7D                    LD   A,L
B463 CD 4B B4              CALL @PRTHX
B466 C9                    RET
B467             ;
B467             @CSR:
B467 F5                    PUSH AF
B468 CD 61 B0              CALL %CSR             ; HL <== YX
B46B 3A D7 B0              LD   A,(@WINXY)
B46E 3C                    INC  A
B46F 95                    SUB  L                ; A - L
B470 2F                    CPL
B471 3C                    INC  A                ; L - A
B472 6F                    LD   L,A
B473 3A D8 B0              LD   A,(@WINXY+1)
B476 3C                    INC  A
B477 94                    SUB  H                ; A - H
B478 2F                    CPL
B479 3C                    INC  A                ; H - A
B47A 67                    LD   H,A
B47B F1                    POP  AF
B47C C9                    RET
B47D             ;
B47D             @SCRN:
B47D E5                    PUSH HL
B47E 3A D7 B0              LD   A,(@WINXY)
B481 3C                    INC  A
B482 85                    ADD  A,L
B483 6F                    LD   L,A
B484 3A D8 B0              LD   A,(@WINXY+1)
B487 3C                    INC  A
B488 84                    ADD  A,H
B489 67                    LD   H,A
B48A CD 64 B0              CALL %SCRN            ; A <== (HL)
B48D E1                    POP  HL
B48E C9                    RET
B48F             ;
B48F             @LOC:
B48F F5                    PUSH AF
B490 E5                    PUSH HL
B491             ;
B491 3A D9 B0              LD   A,(@WIDTH)
B494 3D                    DEC  A
B495 BD                    CP   L              ; X over?
B496 38 16                 JR   C,@LOCE
B498 3A D7 B0              LD   A,(@WINXY)     ; X1
B49B 3C                    INC  A
B49C 85                    ADD  A,L
B49D 6F                    LD   L,A
B49E             ;
B49E 3A DA B0              LD   A,(@MXLIN)
B4A1 3D                    DEC  A
B4A2 BC                    CP   H              ; Y over?
B4A3 38 09                 JR   C,@LOCE
B4A5 3A D8 B0              LD   A,(@WINXY+1)   ; Y1
B4A8 3C                    INC  A
B4A9 84                    ADD  A,H
B4AA 67                    LD   H,A
B4AB CD 67 B0              CALL %LOC
B4AE             @LOCE:
B4AE E1                    POP  HL
B4AF F1                    POP  AF
B4B0 C9                    RET
B4B1             ;
B4B1             @WIDCH:
B4B1             @MON:
B4B1             @BOOT:
B4B1 C9                    RET
B4B2             ;
B4B2             ;**************************
B4B2             ;*      SUBROUTINES       *
B4B2             ;**************************
B4B2             ;
B4B2             ;      Breg. <== WIDTH
B4B2             ;      Creg. <== MAXLIN
B4B2             ;      IX+16 = WINXY
B4B2             ;      AF <== XXX
B4B2             ;
B4B2             WIDLIN:
B4B2 E5                    PUSH HL
B4B3 DD 7E 12              LD   A,(IX+18)
B4B6 21 5C 1F              LD   HL,#WIDTH        ; ﾖｺ
B4B9 BE                    CP   (HL)
B4BA 30 25                 JR   NC,XYERR         ; width over?
B4BC DD 96 10              SUB  (IX+16)          ; A <== X2 - X1
B4BF 38 20                 JR   C,XYERR          ; A < 0 ?
B4C1 28 1E                 JR   Z,XYERR          ; A = 0 ?
B4C3 3D                    DEC  A
B4C4 47                    LD   B,A
B4C5 FE 03                 CP   3
B4C7 38 18                 JR   C,XYERR          ; A < 3 ?
B4C9             ;
B4C9 DD 7E 13              LD   A,(IX+19)
B4CC 21 5B 1F              LD   HL,#MXLIN        ; ﾀﾃ
B4CF BE                    CP   (HL)
B4D0 30 0F                 JR   NC,XYERR         ; line over?
B4D2 DD 96 11              SUB  (IX+17)          ; A <== Y2 - Y1
B4D5 38 0A                 JR   C,XYERR          ; A < 0 ?
B4D7 28 08                 JR   Z,XYERR          ; A = 0 ?
B4D9 3D                    DEC  A
B4DA 4F                    LD   C,A
B4DB FE 02                 CP   2
B4DD 38 02                 JR   C,XYERR          ; A < 2 ?
B4DF             ;
B4DF E1                    POP  HL
B4E0 C9                    RET
B4E1             XYERR:
B4E1 37                    SCF
B4E2 E1                    POP  HL
B4E3 C9                    RET
B4E4             ;
B4E4             ;      CHECK WORK
B4E4             ;      HL    = WORK ADR.
B4E4             ;      Breg. = WIDTH + 2
B4E4             ;      Creg. = MXLIN + 2
B4E4             ;      DE <== NEXT ptr.
B4E4             ;
B4E4             WKCHK:
B4E4 E5                    PUSH HL
B4E5 59                    LD   E,C
B4E6 50                    LD   D,B
B4E7 CD 61 B6              CALL MULDE            ; DE <== D * E
B4EA 19                    ADD  HL,DE
B4EB ED 5B 68              LD   DE,(#WKSIZ)
B4EE 1F
B4EF EB                    EX   DE,HL
B4F0 B7                    OR   A
B4F1 ED 52                 SBC  HL,DE            ; #WKSIZ - WKPTR
B4F3 E1                    POP  HL
B4F4 C9                    RET
B4F5             ;
B4F5             ;      SCREEN ==> WORK
B4F5             ;      BC = (WIDTH+2,MXLIN+2)
B4F5             ;      DE = WORK ADR.
B4F5             ;      HL = (Y1,X1)
B4F5             ;      (BC,DE,HL) <== XXX
B4F5             ;
B4F5             STSCRN:
B4F5 C5                    PUSH BC
B4F6 D5                    PUSH DE
B4F7 E5                    PUSH HL
B4F8             STL1:
B4F8 E5                    PUSH HL
B4F9 C5                    PUSH BC
B4FA             STL2:
B4FA CD 64 B0              CALL %SCRN
B4FD EB                    EX   DE,HL
B4FE CD 9A 1F              CALL #POKE
B501 EB                    EX   DE,HL
B502 13                    INC  DE
B503 2C                    INC  L
B504 10 F4                 DJNZ STL2
B506 C1                    POP  BC
B507 E1                    POP  HL
B508 24                    INC  H
B509 0D                    DEC  C
B50A 20 EC                 JR   NZ,STL1
B50C             ;
B50C E1                    POP  HL
B50D D1                    POP  DE
B50E C1                    POP  BC
B50F C9                    RET
B510             ;
B510             ;      WORK ==> SCREEN
B510             ;      BC = (WIDTH+2,MXLIN+2)
B510             ;      DE = WORK ADR.
B510             ;      HL = (Y1,X1)
B510             ;
B510             LDSCRN:
B510 C5                    PUSH BC
B511 D5                    PUSH DE
B512 E5                    PUSH HL
B513 CD 61 B0              CALL %CSR
B516 22 84 B0              LD   (WORDBF),HL
B519 E1                    POP  HL
B51A E5                    PUSH HL
B51B             LDL1:
B51B E5                    PUSH HL
B51C CD 67 B0              CALL %LOC
B51F 21 86 B0              LD   HL,STRBUF        ; top of line buf.
B522 C5                    PUSH BC
B523             LDL2:
B523 EB                    EX   DE,HL
B524 CD 94 1F              CALL #PEEK
B527 EB                    EX   DE,HL
B528 13                    INC  DE
B529 77                    LD   (HL),A
B52A 23                    INC  HL
B52B 10 F6                 DJNZ LDL2
B52D 36 00                 LD   (HL),0
B52F 21 86 B0              LD   HL,STRBUF
B532 EB                    EX   DE,HL
B533 CD 4F B0              CALL %MSX
B536 EB                    EX   DE,HL
B537 C1                    POP  BC
B538 E1                    POP  HL
B539 24                    INC  H
B53A 0D                    DEC  C
B53B 20 DE                 JR   NZ,LDL1
B53D             ;
B53D 2A 84 B0              LD   HL,(WORDBF)
B540 CD 67 B0              CALL %LOC
B543 E1                    POP  HL
B544 D1                    POP  DE
B545 C1                    POP  BC
B546 C9                    RET
B547             ;
B547             ;      CLEAR IN WINDOW
B547             ;      Acc <== XXX
B547             ;
B547             CLS:
B547 C5                    PUSH BC
B548 D5                    PUSH DE
B549 E5                    PUSH HL
B54A             ;
B54A 11 86 B0              LD   DE,STRBUF
B54D 3A D9 B0              LD   A,(@WIDTH)
B550 47                    LD   B,A
B551             CLS1:
B551 3E 20                 LD   A,' '
B553 12                    LD   (DE),A
B554 13                    INC  DE
B555 10 FA                 DJNZ CLS1
B557 AF                    XOR  A
B558 12                    LD   (DE),A
B559             ;
B559 21 00 00              LD   HL,0
B55C 3A DA B0              LD   A,(@MXLIN)
B55F 47                    LD   B,A
```

▶生協でならOh!Xが490円で買える。大学に受かってよかった！

天見　卓志（18）大阪府

```
B560            CLS2:
B560 11 86 B0           LD   DE,STRBUF
B563 CD 8F B4           CALL @LOC             ; HL ==> YX
B566 CD 4F B0           CALL %MSX
B569 24                 INC  H
B56A 10 F4              DJNZ CLS2
B56C 21 00 00           LD   HL,0
B56F CD 8F B4           CALL @LOC             ; HOME
B572            ;
B572 AF                 XOR  A
B573 21 DB B0           LD   HL,@LCFLG
B576 77                 LD   (HL),A           ; clear
B577 23                 INC  HL               ;   line
B578 77                 LD   (HL),A           ;     connect
B579 23                 INC  HL               ;       flag
B57A 77                 LD   (HL),A
B57B 21 00 00           LD   HL,0
B57E CD 8F B4           CALL @LOC             ; LOCATE 0,0
B581            ;
B581 E1                 POP  HL
B582 D1                 POP  DE
B583 C1                 POP  BC
B584 C9                 RET
B585            ;
B585            ;     LINE CONNECT
B585            ;     Hreg. = LINE
B585            ;
B585            LNCNT:
B585 F5                 PUSH AF
B586 E5                 PUSH HL
B587 CD 96 B5           CALL LCTSB
B58A B6                 OR   (HL)
B58B 77                 LD   (HL),A
B58C E1                 POP  HL
B58D F1                 POP  AF
B58E C9                 RET
B58F            ;
B58F            ;     LINE CONNECT?
B58F            ;     Hreg. = LINE
B58F            ;
B58F            ?LCNT:
B58F E5                 PUSH HL
B590 CD 96 B5           CALL LCTSB
B593 A6                 AND  (HL)
B594 E1                 POP  HL
B595 C9                 RET
B596            ;
B596            ;     H  =  LINE no.
B596            ;     HL <== @LCFLG + ALPHA
B596            ;     A  <== CONNECT BIT
B596            ;
B596            LCTSB:
B596 C5                 PUSH BC
B597 E5                 PUSH HL
B598 24                 INC  H
B599 3E 01              LD   A,01H
B59B            LCTL:
B59B 0F                 RRCA
B59C 25                 DEC  H
B59D 20 FC              JR   NZ,LCTL
B59F            ;
B59F E1                 POP  HL
B5A0 CB 3C              SRL  H                ; H <== H / 8
B5A2 CB 3C              SRL  H
B5A4 CB 3C              SRL  H
B5A6 06 00              LD   B,0
B5A8 4C                 LD   C,H
B5A9 21 DB B0           LD   HL,@LCFLG
B5AC 09                 ADD  HL,BC
B5AD C1                 POP  BC
B5AE C9                 RET
B5AF            ;
B5AF            ;     SCROLL IN WINDOW
B5AF            ;     Acc <== XXX
B5AF            ;
B5AF            SCROLL:
B5AF C5                 PUSH BC
B5B0 D5                 PUSH DE
B5B1 E5                 PUSH HL
B5B2            ;
B5B2 3A D8 B0           LD   A,(@WINXY+1)
B5B5 67                 LD   H,A
B5B6 3A D7 B0           LD   A,(@WINXY)
B5B9 3C                 INC  A
B5BA 6F                 LD   L,A
B5BB 11 86 B0           LD   DE,STRBUF
B5BE 3A DA B0           LD   A,(@MXLIN)
B5C1 3D                 DEC  A
B5C2 4F                 LD   C,A
B5C3 28 1E              JR   Z,SCRL3
B5C5            SCRL1:
B5C5 24                 INC  H
B5C6 24                 INC  H
B5C7 D5                 PUSH DE
B5C8 E5                 PUSH HL
B5C9 3A D9 B0           LD   A,(@WIDTH)
B5CC 47                 LD   B,A
B5CD            SCRL2:
B5CD CD 64 B0           CALL %SCRN            ; A <== char.
B5D0 2C                 INC  L
B5D1 12                 LD   (DE),A
B5D2 13                 INC  DE
B5D3 10 F8              DJNZ SCRL2
B5D5            ;
B5D5 AF                 XOR  A
B5D6 12                 LD   (DE),A
B5D7 E1                 POP  HL
B5D8 D1                 POP  DE
B5D9 25                 DEC  H
B5DA CD 67 B0           CALL %LOC             ; HL ==> YX
B5DD CD 4F B0           CALL %MSX             ; line copy
B5E0 0D                 DEC  C
B5E1 20 E2              JR   NZ,SCRL1
B5E3            SCRL3:
B5E3 24                 INC  H
B5E4 CD F6 B5           CALL LINERA
B5E7 21 DB B0           LD   HL,@LCFLG
B5EA CB 26              SLA  (HL)
B5EC 23                 INC  HL
B5ED CB 16              RL   (HL)
B5EF 23                 INC  HL
B5F0 CB 16              RL   (HL)
B5F2            ;
B5F2 E1                 POP  HL
B5F3 D1                 POP  DE
B5F4 C1                 POP  BC
B5F5 C9                 RET
B5F6            ;
B5F6            ;     1 line erase
B5F6            ;     Hreg. = line no.
B5F6            ;     Acc <== XXX
B5F6            ;
B5F6            LINERA:
B5F6 C5                 PUSH BC
B5F7 E5                 PUSH HL
B5F8 3A D7 B0           LD   A,(@WINXY)
B5FB 3C                 INC  A
B5FC 6F                 LD   L,A
B5FD CD 67 B0           CALL %LOC             ; HL ==> YX
B600 3A D9 B0           LD   A,(@WIDTH)
B603 47                 LD   B,A
B604            LNERL:
B604 CD 43 B0           CALL %PRNTS
B607 10 FB              DJNZ LNERL
B609 E1                 POP  HL
B60A C1                 POP  BC
B60B C9                 RET
B60C            ;
B60C            ;     Window 1 line disp.
B60C            ;     Hreg. = side char.
B60C            ;     Lreg. = space code
B60C            ;     Acc <== XXX
B60C            ;
B60C            LINPRT:
B60C C5                 PUSH BC
B60D D5                 PUSH DE
B60E E5                 PUSH HL
B60F CD 61 B0           CALL %CSR             ; HL <== YX
B612 22 84 B0           LD   (WORDBF),HL
B615 E1                 POP  HL
B616            ;
B616 7C                 LD   A,H
B617 12                 LD   (DE),A
B618 13                 INC  DE
B619 05                 DEC  B
B61A 7D         LNPRL:  LD   A,L
B61B 12                 LD   (DE),A
B61C 13                 INC  DE
B61D 10 FB              DJNZ LNPRL
B61F 7C                 LD   A,H
B620 12                 LD   (DE),A
B621 13                 INC  DE
B622 AF                 XOR  A
B623 12                 LD   (DE),A
B624 D1                 POP  DE
B625 CD 4F B0           CALL %MSX
B628 C1                 POP  BC
B629 E5                 PUSH HL
B62A 2A 84 B0           LD   HL,(WORDBF)
B62D 24                 INC  H
B62E CD 67 B0           CALL %LOC
B631 E1                 POP  HL
B632 C9                 RET
B633            ;
B633            ;     PATCHING S-OS SUB.
B633            ;
B633            PATCH:
B633 3A 7F B0           LD   A,(WINFLG)
B636 B7                 OR   A
B637 C0                 RET  NZ
B638            ;
B638 D5                 PUSH DE
B639 DD 21 15           LD   IX,#TABLE
B63C B0
B63D DD 46 00           LD   B,(IX+0)         ; B <== KAZU
B640 DD 23              INC  IX
B642 21 40 B0           LD   HL,#PATCH
B645            PATL:
B645 23                 INC  HL
B646 DD 5E 00           LD   E,(IX+0)
B649 DD 56 01           LD   D,(IX+1)
B64C 1A                 LD   A,(DE)
B64D 4E                 LD   C,(HL)
B64E EB                 EX   DE,HL
B64F 71                 LD   (HL),C
B650 12                 LD   (DE),A
B651 13                 INC  DE
B652 23                 INC  HL
B653 1A                 LD   A,(DE)
B654 4E                 LD   C,(HL)
B655 EB                 EX   DE,HL
B656 71                 LD   (HL),C
B657 12                 LD   (DE),A
B658 23                 INC  HL
B659 DD 23              INC  IX
B65B DD 23              INC  IX
B65D 10 E6              DJNZ PATL
B65F D1                 POP  DE
B660 C9                 RET
B661            ;
B661            ;     DE <== D * E
B661            ;
B661            MULDE:
B661 E5                 PUSH HL
B662 21 00 00           LD   HL,0
B665 7B                 LD   A,E              ; A <== E
B666 5A                 LD   E,D              ; DE <== D
B667 16 00              LD   D,0
B669            MUL1:
B669 B7                 OR   A
B66A 28 0B              JR   Z,MUL3
B66C CB 3F              SRL  A
B66E 30 01              JR   NC,MUL2
B670 19                 ADD  HL,DE
B671            MUL2:
B671 CB 23              SLA  E
B673 CB 12              RL   D
B675 18 F2              JR   MUL1
B677            MUL3:
B677 5D                 LD   E,L
B678 54                 LD   D,H
B679 E1                 POP  HL
B67A C9                 RET
B67B            ;
B67B            ;***********************************
B67B            ;*      WINDOW PARAMETER AREA      *
B67B            ;***********************************
B67B            ;
B800                    ORG  0B800H
B800            ;
B800            WIB:                                ; WINDOW INF.BLOCK
B800 00 00 00   WNAME:  DEFS 16                     ;   window name
B803 00 00 00
B806 00 00 00
B809 00 00 00
B80C 00 00 00
B80F 00
B810 00 00 00   WINXY:  DEFS 4                      ;   (X1,Y1)-(X2,Y2)
B813 00
B814 00         WIDTH:  DEFS 1                      ;   (X2-X1)-1
B815 00         MXLIN:  DEFS 1                      ;   (Y2-Y1)-1
B816 00 00 00   LCBUF:  DEFS 3                      ;   line connect flag
B819 00 00      XYBUF:  DEFS 2                      ;   cursor locate
B81B 00 00      WKPTR:  DEFS 2                      ;   top of work area
B81D            NEXT:
```



ポケコンの新しい世界

PC-E200/500

山本　信　Yamamoto Makoto

今回シャープから発売された２機種のポケコンPC-E200（22,000円）とPC-E500（28,800円）は，これまでのシャープのポケコン（PC-1200/1300/1400シリーズ）とはまたひと味違う新たなるシリーズ展開を見せています。

まず外見ですが，丸みを帯びたダークグレイのボディはなかなか高級感を持っています。ディスプレイやキーの配列は，1300シリーズと1400シリーズの中間といったところです。どちらもポケットには少々大きめですが（重さはE200が280g，E500が250g），ポケットコンピュータと書いてあるところを見ると，どうやらコートのポケットにでも入れる気でしょうか。

また，中身のほうも従来のものと比べてかなり変化しており，E200とE500との間でもかなり違っています。この２つのマシンは，外観は似ていますがどうやら中身はまったく別のマシンといえそうです。

Z80搭載ポケコンPC-E200

E200の特徴は，まずなんといってもCPUにZ80（CMOS版）を搭載していることです。このため，たとえばX1などでポケコンのプログラムを開発して，RS-232Cで転送して使うなどといったことが簡単にできるようになります。また，BASICにモニタモードがあるので，簡単なものですがマシン語モニタを使うこともできます。ただメモリマップなどが公開されていないので，いまひとつ使い方がはっきりしないのが残念です。早く公開してくれることを期待しましょう。



PC-E200

E200を従来のポケコンと比べてみると，スピードはZ80のおかげもありPC-1475に比べて４倍ほど速くなっています。また，表示能力は24字×４行で，かなりの進歩がみられます。

ソフトウェアの面から見た特徴は，電卓モードがないことでしょう。最近のシャープのポケコンには電卓モードが付いているものが多かったのですが，このマシンではあえてそれを付けていないようです。実際，ポケコンを使うときは電卓モードではなく，計算式を見たあとから訂正ができる，RUNモードを使うことが多いでしょう。ポケコンにはRUNモードとPROモードとがあり，RUNモードではプログラムを打ち込めないので，数字を直接打ち込んでも行番号と間違えられることがありません。

また，このときラストアンサ機能や，CONST機能を使うことができます。CONST機能というのは，ある演算（４を足すとか３を掛けるなど）を定義すれば，それを自動的に実行してくれる機能で，同じ計算を繰り返し行うときに便利です。

このほか86種類の関数が使え，２変数統計もできるといった点で関数電卓としてもなかなかのものだといえるでしょう。ただし，統計計算ではデータをあとから確認できないので少々不便ですが，それほどデータが多くなければたいして困ることはありません。

BASICは，従来からのシャープのポケコンBASICの拡張ですが，PC-88/98のN88-BASICに似た感じで拡張が行われています。これは移植性を考慮したものでしょう。もちろん従来機（PC-1400シリーズ）のプログラムをテープから読むこともできます。また，RAMをプログラムファイルとして使用できるので，使わないプログラムをしまっておくことが可能です。通常は外部記憶を持っていないポケコンにとって，この機能は必需品といえるでしょう。

また，TEXTキーを押すとテキストエディタモードに入ります。このモードで，RS-232Cを通じ，BASICプログラムをパソコンとの間で受け渡しすることができます。

さらにCASLモードというものもあります。これは，情報処理技術者試験で出題される仮想コンピュータCOMETのアセンブラCASLの学習用に作られたものです。先述のテキストエディタモードで書いたプログラムを，CASLシミュレータで実行することができます。電車のなかや学校など，ちょっとした時間を使って情報処理技術者試験の勉強をするには便利でしょう。

ミニワークステーションPC-E500

E500は，ポケコンというよりもポケットワークステーションとでもいうべきものです。CPUはZ80ではないようですが，E200よりもさらに速く，PC-1475の７倍速となっています。また，表示能力も文字40字×４行，グラフィックは240×32ドットというスペックを誇ります。

しかし，E500のもっと凄いところはそのソフトウェアです。メニューキーを押すとメニュー画面になり，BASIC, CALC, MATRIX, STAT，エンジニアの各メニューをファンクションキーで選択するようになっています。

まずBASICですが，これはE200のBASICをさらに強化したものです。もちろん従来機のプログラムも読めますが，かなり強力なBASICとなっていて，まるでパソコンのようです（まるで98のようだという話も



PC-E500

ある)。

また，E500のBASICはOSとしての機能も持っています。たとえば，本体内のRAMの一部をRAMディスク（RAMファイルという）として使うことができ，また64Kバイトまでの増設RAMカードもRAMファイルとして使えるので，RAMカードをフロッピーディスク感覚で使うことができます。さらに，外付けの2.5インチFDD(CE-140F)を使うことも可能です。そして，BASICにフォーマットやファイルコピーなどの命令があるので，いちいちユーティリティを使ったりしないでいいところなどは，パソコンをも上回っているといってもいいでしょう。

ほかにも，倍精度数値変数（仮数部20桁指数部2桁）が使用でき，テキストエディタモードもあるのでパソコン用のプログラムを書いて，RS-232Cで転送するといったこともできます。さらに，AER（数式登録機能）というものもあります。よく使う公式なのだがBASICでプログラムを組むほどではないといった場合，関数を作って登録しておけば，さっと呼び出して計算することができます。この関数はBASICプログラム内からも使うことができます。

次にCALモードですが，このモードでは普通の関数電卓として使用できます。数式通りに計算してくれるので，たとえば“1＋1＊2”は4ではなく3になります。メモリも普通の電卓のメモリのほかにA～Z（A$～Z$）までのメモリが使え，これはBASICと共通なのでCALモードとBASICともデータ交換をすることができます。また，もちろんE200のようにBASICのRUNモードで計算することもできます。私は式が見えて便利なのでRUNモードのほうをよく使っています。

行列の計算をするMATRIXモードでは，メモリが許す限り任意の大きさの行列を使用でき，連立1次方程式を解くこともできます。また，BASICの配列変数を使ってのデータのやりとりや，BASICから行列演算機能を呼び出すこともできます。

STATモードは統計計算のモードです。このモードでは2変数統計も扱えます。データも保存されるので修正や追加もでき，たいへん便利です。また，データや結果をBASICとやりとりすることができ，さらに，BASICから統計計算機能を使うこともできます。

便利なエンジニアモード

E500にはエンジニアモードが付いています。この機能はいままで紹介してきた各モードとは若干異なり，いわば実用プログラム集がROMで入っているといったものです。このプログラムはエンジニアリングソフトウェアと呼ばれ，エンジニアのメニューを選択すると，数学，科学，工学，統計，編集のメニューが現れます。たとえば，数学ならニュートン法による方程式を解くプログラムが使え，科学では原子の周期表や物理定数の一覧が，また，工学では複素数の計算など，全部で35個のエンジニアリングソフトウェアをメニューで選択して使えるわけです。

このモードのさらに凄いところは，メニューをユーザーが変更できることと，ユーザーが作ったBASICプログラムをRAMファイルにセーブしてメニューに登録すれば，最初から持っているソフトと同じようにメニューから呼び出すことができるという点です。この機能を使って，ポケコンを自分用にカスタマイズすれば，まさにポケットワークステーションとなるでしょう。

ところで，メニューをよく見ると“デモ”という項目もあり，見てみるとデモプログラムが入っていました。

このほかE500は細かいところがよくできています。キーボードはロールオーバー(前のキーを離す前に次のキーが押せる）が可能になっていて，ちゃんとキーバッファも付いています。また，CTRL-Oでキークリックが ON/OFFできるといった点も，ポケコンのキーは感触が頼りないので，ありがたいところです。このE500は，まさに使うために作られたマシンだといえるでしょう。

もう少し工夫のほしいマニュアル類

両機種とも同じようなマニュアルが付いてきますが，受ける感じはかなり違います。

E200のほうは，初心者でもわかるように書かれています。これはE200がモードを減らした分，わかりやすいポケコンであるといったことにもよるのでしょう。

一方，E500のマニュアルは，一見まとまっているような印象を受けますが，読んでみると，なんでもかんでも詰め込まれているような感じがします。これだけの機能があるのですから仕方がないともいえますが，これまでポケコンやパソコンを使ったことのある人でないとわかりにくいようなところがあります。

両方ともBASICのリファレンスマニュアルはコマンドがアルファベット順に並んでいますが，これは機能別に分けたほうがよかったでしょう。また，予約語の一覧がないというのも困ったものです。しかし，最大の欠点は索引がないということです。昔ならそれでもよかったかもしれませんが，これだけ機能が増加しているのですから早くなんとかしてもらいたいところです。

Eシリーズの将来に期待

今後シャープのポケコンのうち，PC-1300/1400シリーズはEシリーズとして発展していくものと思われます。一方，PC-1200シリーズはワイシャツの胸ポケットにはいるという小ささを生かして，入門用，または，業務専用ポケコンとして発展していくのではないでしょうか。また，シャープといえば最近電子手帳が話題になっていますが，これがポケコンと結びついたら面白いことになるでしょう。ともかく，E500の愛用者カードに書かれている項目などを見ても，シャープのこのシリーズに対する力の入れようがわかります。今後のポケコンからは目が離せないといえそうです。

PC-E200　22,000円
PC-E500　28,800円
ポケットディスクドライブCE-140F(オプション)　49,800円
シャープ　☎06(621)1221, 03(260)1161

愛読者プレゼント

●
プレゼントの応募方法

とじ込みのアンケートはがきの該当項目をすべてご記入のうえ，希望するプレゼント番号をはがき右下のスペースにひとつ記入してお申し込みください。締め切りは1988年7月18日の到着分までとします。当選者の発表は1988年9月号で行います。

1
日本ファルコム　☎0425(27)6501



ソーサリアン
X1turboシリーズ専用
(model10は要CZ-8BGR2, CZ-8BF1)
5"2D版　9,800円
2名

ファンタジーファン待望のソーサリアンX1turbo版。はやる気持ちを抑えられない向きには，「とにかく早く始めたい人のためのインスタントパーティー」が用意されている。

2
ゲームアーツ　☎03(984)1136

ゼリアード



X1turboシリーズ
2ドライブ専用
(model10では動作しません)
5"2D版　7,500円　2名

ファンタジーロールプレイングの2本目はゼリアード。姫君と王国を窮地から救うために怪物たちと闘うヒーロー。アニメがなかなか芸が細かい。

3
工画堂スタジオ　☎03(353)7724



a.アルギースの翼
X1turboシリーズ
2ドライブ専用
(ただしX1でもFM音源ボードがあれば動作可能)
5"2D版　7,800円　3名

今月のファンタジーゲームの締めはアルギースの翼。中世を思わせる惑星を舞台にしたSF・RPG。戦闘シーンが見ものです。

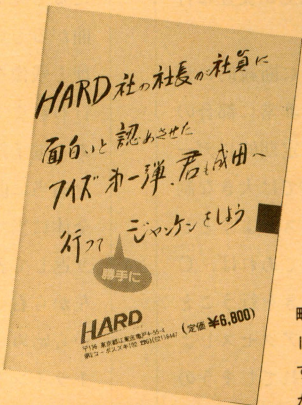
b.オリジナルディスクケース　10名



工画堂スタジオのロゴマーク入りディスクケースを10名に。色はグレー。

4
ハード　☎03(621)8447

ハード社の社長が社員に面白いと認めさせたクイズ第1弾，君も成田へ行って勝手にジャンケンをしよう



X1/X1turboシリーズ用
5"2D版　6,800円
10名

略称「勝手に成ジャン」ゲームを10名の読者に。クイズ1問を正解するごとに女の子のグラフィックが出てくるんだって。ウキウキ。

5月号プレゼント当選者

1　桃太郎伝説（東京都）安井研二（静岡県）砂子満　2　紫醜羅（大阪府）松野康利（愛媛県）兵頭直樹　3　麻雀狂時代 SPECIAL（群馬県）成川浩一（埼玉県）福辺徳明　4　コンピュータ言語進化論（宮城県）宗片陽一（静岡県）後藤智明（三重県）西川正哉

以上の方々が当選されました。おめでとうございます。品物は順次発送いたしますが，入荷状況などにより遅れることがあります。また，公正取引委員会の告示により，このプレゼントに当選された方は，この号の他の懸賞には当選できない場合がありますのでご了承ください。

QUESTION and ANSWER

Q **X68000のスプライトコントローラには構造的欠陥があるのではないでしょうか。画面の上のほうだとやたらチラつきます。付属のグラディウスでさえそうです。早急に調査をお願いします。**　**東京都　鹿島　真一**

A ハードの制約により，正しくないプログラムでは鹿島さんのおっしゃる症状は起こりえます。が，グラディウスが上記に当てはまるとも思えず，実際，チラつきを確認することはできませんでした。そこで，ここでは鹿島さんのハードの問題かもしれないと解答したうえで，一般的な話をしておきます。

スプライトを表示するには，まず，CPUが，スプライトコントローラのレジスタに表示する位置やパターン番号を書き込み，このデータとスプライトパターンをCRTCが読み出して画面に表示するという2つの段階があります。ここで注意しなければならないのは，CPUがデータを書き込むのと，CRTCがデータを読み出すのを同時に行うわけにはいかないということです。そのため，スプライトコントローラへのアクセスは図1のようにCPUとCRTCが交互に行うようになっています。

この仕組みはCRTCのように断続的にデータを読み出す側にとっては非常に都合のよい方法です。ところが，不定期的にデータを読み書きするCPUにとっては大きな制約となります。CPUがアクセスしようとしたときが，図1の(a)の時点であれば，CPUはすぐにデータの読み書きを行うことができますが，(b)のときにアクセスしようとすると，(c)まで待たされてしまうのです（ウエイトが入るという)。この待たされている間，CPUはなにもすることができません。ゲームのように速度が要求される場面では，この金縛りにあっている時間は無駄以外のなにものでもないわけです。

これを防ぐために，スプライトコントローラへのアクセス期間をCPUが独占できるような方法が用意されています。邪魔者がいなくなりますから，CPUは高速にデータの読み書きができることになります。ところがこの状態では今度はCRTCがアクセスできなくなってしまいますね。CRTCがアクセスできないということは画面になにも表示されないということ，つまり，スプライトの表示がすべてカットされてしまうということです。この2つのこと，「スプライトコントローラへのアクセス時にはウエイトが入ることがある」，「ウエイトが入らないようにするとスプライトが表示されなくなってしまう」が，最初に述べた制約に当たります。

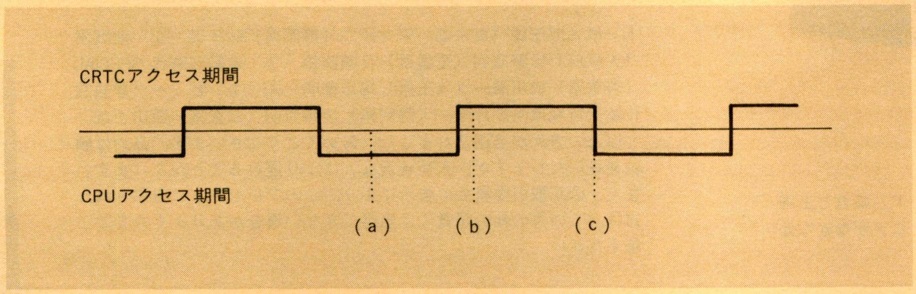
では，どうしたらよいかですが，その前に少しCRTディスプレイの原理についての話をしておきたいと思います。CRT（ブラウン管）の基本原理は，電子が蛍光体に当たると光るということと，電磁界に応じて曲がるというごく単純なものです。この原理と残像を利用してブラウン管での表示が行われます。具体的にはブラウン管の奥にある電子銃から放たれた1本の電子ビームを適当に曲げて(偏向させて)，順に蛍光面全体に当てる（走査する）ということを繰り返して表示を行っています。この順序は左から右へ，そして上から下へです。左上から始めて右上まで走査したら左へ戻り，1ライン下を走査します。一番下のラインの走査が済んだら，また左上に戻って走査を続けます。ある瞬間に光っているのは電子ビームが当たっているただ1点ですし，右から左へ戻るとき（水平帰線期間）と，下から上へ戻るとき(垂直帰線期間)にはディスプレイにはなにも表示されていませんが，残像が残りますので我々の目にはちゃんと1画面が表示されているように見えるわけです。

さて，このあたりでうまいぐあいにスプライトを表示するにはどうすればよいかが見えてきました。「画面になにも表示されていない水平帰線期間ないしは垂直帰線期間にスプライトの表示をカットして一気にデータを書き換える」のです。実際には水平帰線期間にはデータを書き換えるだけの十分な時間がありませんので，主に垂直帰線期間が利用されます。この方法はシャープも推奨しているもので，解析したわけではありませんが，グラディウスもこのようにしてスプライト表示を行っているものと思われます。もうおわかりのように，垂直帰線期間内でデータの書き換えを終えることができず，一瞬でもスプライトの再表示が遅れてしまえばチラついてしまうことになります。多量のスプライトを扱うゲームプログラムではありうることです。

なお，スプライトの数があまり多くなくても，画面の上がチラつくことがあります。これは垂直帰線期間の検出方法がまずいためです。正しいプログラムでは垂直表示期間から垂直帰線期間への変わり目を検出するべきで，これなら垂直帰線期間をフルに使うことができるわけですが，「今が垂直帰線期間かどうか」だけを調べるようなプログラムでは，もしかすると垂直帰線期間が終わる寸前かもしれませんので時間が足りなくなってしまうのです。ただ，これも善し悪しで，単に垂直帰線期間になるまで待つときの待ち時間は最大でも「垂直表示期間分」ですが，変わり目を検出するには最大で「垂直表示期間＋垂直帰線期間」待たなければなりません。

なお，割り込みを利用してロスタイムなしに垂直帰線期間への変わり目を知るスマートな方法もあります。　　（村田　敏幸）

図1　スプライトコントローラへの時分割アクセス



Q S-OSのアスキーファイルに関して質問があります。X1のHuBASICのファイル管理とS-OSのそれとは同一であるということになっていますが，HuBASIC上で作ったアスキーファイルがS-OS上で（E-MATEなどを使って）読み込めません。エンドコードを$1A_H$から00_Hへと変更するだけではだめなのでしょうか。　三重県　小松　英生

A X1のHuBASICとS-OSのファイル管理の方法が同じであるというのは，ディレクトリテーブル(ファイルの情報を記録しておくところ）やFAT（ファイルの大きさやつながり方を記録しておくところ）の位置が共通であるということです。

しかし，いくらファイル管理の方法が同じでも，ファイル自体が異なれば当然お互いに読み書きすることはできないのです。

ファイル自体が違うというのはどういうことなのでしょうか。これは同じ内容のファイルでもHuBASICとS-OSでは表記の方法が違うということなのです。つまり，S-OSでのアスキーファイルは属性がAscとなっているだけで実質はマシン語ファイル（Bin）とまったく一緒になっています。

すなわち，ディレクトリテーブルにアスキーテキストの開始番地，サイズを書いておくことによってファイルの大きさを知るようになっているのです。

一方，HuBASICではアスキーファイルの大きさはFATを使って計算されます。すなわち，FATを見て何クラスタ，何レコード使っているかを調べ，それをもとにしてファイルの大きさを出すのです。また，ディレクトリテーブルの開始番地やサイズを書き込んでおく部分には，常に0000_Hが書き込まれます。

少し脱線しますが，このようなファイルの表現法の違いはそれぞれのシステムにおけるアスキーファイルの存在価値の違いによるところが大きいのです。というのも，S-OSではアスキーファイルといえばテキストですから，処理が単純なほうがよいし，基本的にオンメモリで考えればよいのでサイズが64Kバイト取れれば十分なのです。そこで，Binファイルと同じようなフォーマットをとっているわけです。

一方，HuBASICではアスキーファイルはデータファイルとして扱うことが多いので，ファイル長を64Kバイトに制限されると不便なのです。そこでFATでファイル長を自由に設定できるようになっているわけです。

さて，S-OSとHuBASICのアスキーファイルの違いは，ファイルの大きさの表現法の違いだけではありません。ファイルのエンドコードも異なっているのです（これは小松さんも試してみられたようですが）。すなわちHuBASICのアスキーファイルのエンドコードは$1A_H$であるのに対し，S-OSでのアスキーファイルのエンドコードは00_Hです。

ところで，マシン語ファイルはHuBASICとS-OSで互換性があります。これはもうおわかりのようにHuBASICとS-OSでマシン語ファイルの表記がまったく共通なためです。ところがアスキーファイルでは共通でないためにファイルのやり取りができないのです。

次に，理由がわかったところでどうすればよいかを考えてみましょう。エンドコードを書き換えただけでは，もちろんだめです。そもそもファイルの大きさを知ることができないためにエンドコードにたどり着くことすらできないからです。

ではどうするのかというと，エンドコードを書き換える以外にも，ファイルの大きさの情報を付け加えなければならないのです。S-OSからHuBASICに移すときは簡単です。エンドコードを00_Hから$1A_H$に書き換え，さらにディレクトリテーブルの開始番地，サイズの部分を0000_Hで埋めればHuBASIC式のアスキーファイルの出来上りです。

逆にHuBASICからS-OSへと移すときは少し面倒になります（といってもそれほどではありませんが）。エンドコードを$1A_H$から00_Hに書き換えるのはもちろんですが，さらにFATからファイルのサイズを計算し，ディレクトリテーブルのサイズの部分に書き込まなければならないのです（開始番地は0000_HのままでOK）。

FATに関する詳しい説明は文献を読んでいただきたいのですが，FATからファイルのサイズを計算するにはそのファイルが全域を使用しているクラスタ数a，レコード数bを求め，さらにファイルの最終レコードを読み込んでエンドコードを探すことによって最終レコード内で使っているバイト数cを求め，

$$a \times 4096 + b \times 256 + c$$

とします。

たとえば，HuBASIC上のあるアスキーファイルが，03_H～10_Hクラスタおよび11_Hクラスタの$0A_H$レコードまでを全域使用し，$0B_H$レコードの第80_Hバイトで終わっているとすると，

$a = 0C_H$

$b = 0B_H$

$c = 81_H$

となりますから，ファイルのサイズとして$CB81_H$を書き込むことになります。

以上が基本的なファイルコンバートの方法ですが，うまくやるとひとつのファイルでHuBASICでもS-OSでも読めるようにすることも可能です。　（華門　真人）

質問にお答えします

日ごろ疑問に思っていること，どんなことでも結構です。どんどんお便りください。難問，奇問，編集室が総力を上げてお答えいたします。ただし，お寄せいただいているものの中には，マニュアルを読めばすぐに回答が得られるようなものも多々あります。最低限，マニュアルは熟読しておきましょう。質問はなるべく具体的に機種名，システム構成，必要なら図も入れてこと細かに書いてください。また，返信用切手同封の質問をよく受けますが，原則として，質問には本誌上でお答えすることになっていますのでご了承ください。なお，質問の内容について，直接問い合わせることもありますので，電話番号も明記してくださいね。

宛先：〒102 東京都千代田区
九段南2-3-26井関ビル
㈱日本ソフトバンク出版部
「Oh！X質問箱」係

FILES Oh!X

このインデックスは，タイトル，注記――筆者名，誌名，月号，ページで構成されています。ここでもすっかりX68000のパートが増えました。

参考文献
I/O　工学社
ASCII　アスキー
Hacker　日本文芸社
パソコンワールド　コンピュータワールド・ジャパン
POPCOM　小学館
マイコン　電波新聞社
マイコン BASIC Magazine　電波新聞社
LOGIN　アスキー

一般

▶ Big New Products　第3世代のマイクロプロセッサM88000
日本モトローラのRISCマイクロプロセッサM88000について。――編集部，I/O，6月号，252-253pp.
▶ ASCII EXPRESS 21世紀の通信環境「ISDN」サービスがスタート
NTTが開始した，ISDNに基づくサービスINSネット64について。――編集部，ASCII，6月号，162p.
▶ ASCII EXPRESS　シャープ，名前・電話番号を記憶する電子メモを発売
電子メモPA-370について。――編集部，ASCII，6月号，166p.
▶ PRODUCTS SHOWCASE　シャープ CZ-8NSI
A4判カラーイメージスキャナCZ-8NS1の機能や使用感などをグラビアで紹介。エプソンGT-3000のコマンドとの比較もしている。――編集部，ASCII，6月号，248-249pp.
▶シャープが関数ポケコン2機種を発売
基本機能が充実しているPC-E200と，高機能を追求したPC-E500の性能を紹介。――編集部，POPCOM，6月号，151p.
▶ MicomNews　鮮やかなカラー印刷を高速で　カラーインクジェットIO-730/735
シャープから新発売されたカラーインクジェットプリンタの機能，価格についての紹介。――編集部，マイコン．6月号，209p.

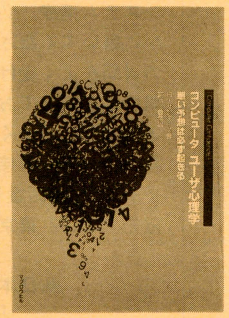
MZ-80K/C/1200/700/1500

MZ-80K/C/1200/700/1500
▶ SKATE GAME
できるだけ少ない歩数で，リンクを5周するというスケートゲーム。MZ-1500の場合はSP-5030 BASICが必要。――こっくりさん，マイコンBASIC Magazine，6月号，141p.
MZ-700/1500
▶ Hu-TYPE
横スクロールシューティングゲーム(HuBASIC)。ピョンピョン飛び跳ねるUFOを操作して，敵をやっつけろ。――MBA，マイコンBASIC Magazine，6月号，144-145pp.
▶ BLOCK CITY
敵キャラを利用して，ブロックを壊すゲーム（S-BASIC)。全13面。――カリット，マイコンBASIC Magazine，6月号，142-143pp.
MZ-700
▶誌上公開質問状　シャープ（MZ）
MZ-700のBASICのコピーのしかたについて。――マイコンBASIC Magazine，6月号，71-72pp.
MZ-1500
▶木の精
木の妖精を森の中の木から木へと移してあげよう。パズル性のあるアクションゲーム。――あむーる，マイコンBASIC Magazine，6月号，146-147pp.

MZ-80B/2000/2500/2800

MZ-80B/2000/2500/2800
▶ DRAGON BUSTARD
6匹の竜と魔術師を倒して，さらわれた姫を助け出すゲーム。――大井隆広，マイコンBASIC Magazine，6月号，148-149pp.
MZ-2200/2500
▶ MZ版ビルこわし
ブルドーザに乗って，ダイナマイトをうまく操作しビルを壊していくゲーム。FMからの移植版。――Pchan マイコンBASIC Magazine，6月号，150-152pp.
MZ-2500
▶パソコン活用テクノロジー　スーパーMZでWD文書ディスクをアナライズする
MZ-2500とWDシリーズ相互間の文書コンバートについて。――有沢公明，Hacker，6月号，39-46pp.
▶ぽぷこみかる☆まあじゃん
ポプコム編集部スタッフを相手に繰り広げられる麻雀ゲーム。――ORESAMA，POPCOM，6月号，271-284pp.
▶なんでもQ&A　シャープMZシリーズ編
イメージ情報ステーションMZ-1V01を使用するパソコンファクスのサポートについて。――シャープ，マイコン，6月号，442-443pp.
▶誌上公開質問状　シャープ（MZ）
MZ-2500シリーズ用の活用本の紹介など。――MZいちばん，マイコンBASIC Magazine，6月号，70-71pp.
▶ CRYSTAL
敵の動きを見極めて，クリスタルを集めるアクションゲーム。――蒲生敬，マイコンBASIC Magazine，6月号，153-155pp.
MZ-2861
▶シャープMZ-2861用統合OAソフトウェアupシリーズ使用レポート　（DTPを意識した）デスクup
upシリーズの紹介で最後の1本となったワープロソフト，デスクupの解説。――編集部，マイコン，6月号，374-379pp.
▶なんでもQ&A　シャープMZシリーズ編
MZ-2861用のアプリケーションソフトupシリーズ4種についての解説。――シャープ，マイコン，6月号，443p.

新刊書案内



この本を簡単に言ってしまえば「心理学者によって書かれた，オフィスにおけるコンピュータを取り巻く心理」の本です。コンピュータを直接取り扱う人々は，なんでも数字に置き換えたがる傾向があるが，実際の経営の意志決定のための要素は数字では置き換えることができないことが多く，むしろ置き換えると，逆に弊害を引き起こす可能性がある（数字だけによる管理は，組織を自滅させることができる）とか，オペレータとエンドユーザーは，視点がまったく違うのですれ違いを起こすことが多いなど，コンピュータを前にしたときの人間について書かれているのです（コンピューター犯罪に関する考察もあります)。個々の内容はそれなりに価値があるのですが，翻訳のせいなのか，もともとこういう本なのか，全体としてはかなり読みづらい文章です。少なくとも良い翻訳とは言えないでしょう。が，このたぐいの本はまだ数が少なく，また，最近はコンピュータが広く使われるようになったことで，ますます大きなテーマとなりつつあります。我慢しても読む価値がある本なのかもしれません。（た）

コンピュータ　ユーザ心理学　B・サンダース著
菊池豊彦訳　マグロウヒルブック刊
B6判　144ページ　1,900円　☎03(542)8821

X1/X1turbo/Z

X1シリーズ

▶ EX-MONITOR VI.0
X1用高機能，高速マシン語モニタ。ヘルプメッセージも出る。——謁宮守一，I/O，6月号，204-207pp.

▶誌上 RPG サンダーロード
テキスト RPG 第3章，勇者のヨロイ。——グループ・クラムボン，POPCOM，6月号，240-249pp.

▶ NEBA ACT II
迫りくるインベーダーを倒し，全30面クリアを目指す。——西嶋茂，POPCOM，6月号，252-258pp.

▶誌上公開質問状　シャープ（X1）
X1につなげられるフロッピーディスクや拡張I/Oボックス，ディスプレイなどについてのごく簡単な解答。——多田太郎，マイコンBASIC Magazine，6月号，73-74pp.

▶ ZILLION
横スクロール　シューティングゲーム。J.J.ZILLIONで敵を撃破せよ。——A．岩ちゃき，マイコン，6月号，227-284pp.

▶ THE へのへのもへじ
鉛筆君を動かして，へのへのもへじを画面に作るという。パズルゲーム。——下中順司，マイコンBASIC Magazine，6月号，194-195pp.

▶ TACO-SUBMARINE
シューティングゲーム。T国の潜水艦の乗組員となって，O国の航空部隊を全滅させよう。——えばちょう，マイコン BASIC Magazine，6月号，196-198pp.

X1turbo シリーズ

▶ X1turbo IIに6MHz ターボチャージャーを！
turbo IIの基板のパターンカットなどを行い，turbo IIのクロックを6MHz にアップさせる。——今雪　寛，I/O，6月号，123-126pp.

▶なんでもQ＆A　X1/X1turbo/X68000シリーズ編
X1turboZ II (New Z-BASIC) でのクロマキー合成について。BASIC でのサンプルプログラム付き。——シャープ，マイコン，6月号，440-441pp.

X68000

▶メディア・エディタ68K
X68000の5"2HD用モニタ，エディットプログラム。——KENYA，I/O，6月号，195-199pp.

▶ TIMER-D を使ったシステムダウンプログラム
タイマ割り込みDを使い，一定時間後にシステムダウンさせる迷惑なプログラム。タイマの使い方の参考にはなるが。——高橋純，I/O，6月号，254-256pp.

▶ ASCII EXPRESS　C&B，パソコン・ビデオ対応の X68000用ソフトを発売
パソコンに接続可能なS-VHSビデオ「Com・Vi」とX68000を使用してCGアニメの合成などができる「Hyper UD Com•Vi リンク」について。——編集部，ASCII，6月号，172p.

▶ ASCII EXPRESS　イーストが X68000用の日本語ワープロソフトを発売
X68000用ワープロソフト「EW」について。——編集部，ASCII，6月号，173p.

▶失敗しないハードディスク選び　ハードディスク製品一覧
主に98用に発売されているハードディスクの紹介だが，X68000に使う場合の注意も簡単に述べている。——編集部，ASCII，6月号，199p.

▶ X68000　WORKSHOP
日本語ワードプロセッサ EW を紹介。カナ漢字サブルーチンの特徴についても述べられている。——編集部，ASCII，6月号，277-278pp.

▶ X68000　WORKSHOP　X68K　Programmer's Shop
今回はスプライトを表示するためにビデオコントローラレジスタや画面モードレジスタなどについて解説。——編集部，ASCII，6月号，279-284pp.

▶国内ニュース New'88JUNE.X68000対応「Video 工場」
シー・アンド・ビーから発売されたビデオ編集システム Video 工場についての簡単な紹介。——編集部，パソコンワールド，6月号，174p.

▶話題の新機種リポート
X68000ACE，ACE-HD の X68000からの変更点をソフト，ハードの両面から紹介。——編集部，POPCOM，6月号，154-156pp.

▶ X68000マシン語入門
「必須」命令のしめくくりとして，tst，neg，clr，lea，trap 命令などについて解説している。——高橋雄一，マイコン，6月号，200-208pp.

▶ Micom News　X68000用ワードプロセッサ EW
イーストより発売になった X68000用ワープロ EW の機能，価格についての紹介。——編集部，マイコン，6月号，214p.

▶ CG ツール使いこなしシリーズ第2回
Z'sSTAFF PRO-68K の使いこなしシリーズ第2回。先月のペン機能の解説の続き。——紀要介，マイコン，6月号，294-297pp.

▶なんでも Q&A　X1/X1turbo/X68000シリーズ編
X68000の THE 福袋 V2.0の内容について。——シャープ，マイコン，6月号，440p.

▶なんでも Q&A　X1/X1turbo/X68000シリーズ編
X68000ACE-HD でハードディスクの設定をしてからゲームを立ちあげる方法について。——シャープ，マイコン，6月号，441p.

▶なんでも Q&A　X1/X1turbo/X68000シリーズ編
X68000用ミュージックソフト MUSIC PRO-68K で大きな楽譜を入力する方法について。——シャープ，マイコン，6月号，441p.

▶誌上公開質問状　シャープ
X68000に NEC のプリンタを接続する方法について解説している。——多田太郎，マイコン BASIC Magazine，6月号，74p.

▶ BALL BALL
2人で対戦できるバレーボールゲーム。ジョイスティックが2つ必要。——村崎裕一，マイコンBASIC Magazine，6月号，199-200pp.

▶ A-JAX “FIGHTING SPIRIT”
ゲームミュージックプログラム。——Yu-You，マイコン BASIC Magazine，6月号，206-209pp.

▶ A-JAX “RANKING”
ゲームミュージックプログラム。——川野俊充，マイコン BASIC Magazine，6月号，210-212pp.

▶チャレンジ！ X68000
X68000の新作ゲーム，ドラゴンスピリット，R-TYPEの開発状況を紹介。——川野俊充，マイコンBASIC Magazine，6月号，297-298pp.

▶ X68000新聞
完成間近のサンプリングソフト Sampling PRO-68K を紹介。——編集部，LOGIN，6月号，230-231pp.

ポケコン

▶ JEWEL THIEF
名ドロボウになって防犯カメラに見つからないように宝石を盗み出すゲーム。——江波戸篤人，マイコンBASIC Magazine，6月号，203p.

PC-1416G/1417G

▶誌上公開質問状　シャープ（ポケコン）
PC-1417G でオリジナルキャラを表示させる方法とアフターサービスの連絡先。——OGI，マイコン BASIC Magazine，6月号，72-73pp.

PC-1445/E200

▶ CASL 太鼓判　第6回
CASL の命令解説6回目。今回はスタック操作，コール，リターンなどのスタック関係命令について。4月17日に実施された第2種情報処理技術者試験の CASL に関する問題の一部の解説もある。——塚田洋一，マイコン，6月号，402-405pp.

PC-1501

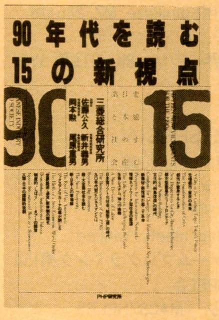
▶ DOG FIGHT
タイトルからも知れるように，ポケコン版アフターバーナー（？）のゲーム。——山沢正裕，マイコン BASIC Magazine，6月号，204p.



コンピューターが世界を変える　3

NHK 特集のシリーズ第3巻は，「明日のコンピューターに何を求めるのか」と銘打って，現在の先端技術がどんなふうに世界中で利用されているのかを紹介する。米国防総省と NASA が開発に取りくんでいる超々音速機ニュー・オリエント・エキスプレスや，フランスで普及中のネットワーク，ミニテルなどを始め，明るい未来がたくさん描かれている。クレイ2や最新鋭戦闘機の写真が少々情けないのが残念だ。

NHK 取材班編　角川書店刊
A5判　234ページ　1,600円　☎03(238)8521



90年代を読む15の新視点

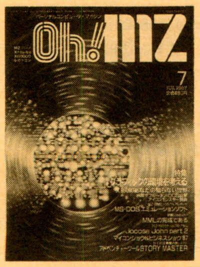
本書は，目前に迫った1990年代を，都市と社会，産業とテクノロジー，そして国際関係などの視点から予測しようと試みている。4人の著者はいずれも三菱総合研究所のメンバーだ。都市構造に関する論議を始め，バイオテクノロジー，大企業時代からの変化，世界経済の見通しなどについて語られている。が，惜しいことに「1990年代のコンピュータ」については触れられていない。

佐藤公久　新井義男　岡本勲　尾原重男　共著
PHP 研究所刊
A5判　254ページ　1,300円　☎03(239)6221

BACK ISSUES バックナンバー案内

ここには1987年7月号から1988年6月号までをご紹介しました。現在，1987年2,3,4,5,7,8,9,10,11,12，1988年1,2,3,4,5,6までの在庫がございます。バックナンバーおよび定期購読のお申し込み方法については，本文168ページを参照してください。

1987



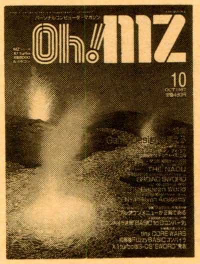
7月号
特集　グラフィックの環境を考える
MZ-2500とサポート/ビジュアルマシンとしてのX1
THE SOFTOUCH　キングス・ナイト・スペシャル
魔界復活/三国志/新作情報他
X68000あなたの知らない世界　内部サブルーチンIOCS
●MZ-2861のMS-DOSとエミュレーションソフト
●MZ-1500用投稿ゲーム　Jocose John part2
全機種共通システム　アドベンチャーゲーム作成ツールSTORY MASTER



8月号
特集　迷宮の日本語処理環境
MZ-2500用ワープロプログラムSuperものかきくん
書式ユーティリティCOLN/らくらくSYMBOL他
試験に出るX1　最終回　通信プログラムである
X68000BASIC入門　第1回　めぐりあいX-BASIC
●X1/turbo用パズルゲーム　STAR PANIC
●Z'sSTAFF PRO 68Kの世界
X68000あなたの知らない世界　SOUND PRO-68K他
全機種共通システム　FM-7/77版S-OS"SWORD"他



9月号
特集1　MZ-700に不可能はない
MZ-700ゲームテクニック集/SPACE BLUSTER SG
特集2　ミュージックデータと遊ぶFM音源の世界
MZ-2500MMLの拡張/X1/turbo用MMLコンバータ
X68000あなたの知らない世界　マシン語入力ツール
BASICリレー連載　ディレクトリまるごとコピー
●X1turboZ，X68000用ハードコピープログラム
全機種共通システム　PC-80/88版S-OS"SWORD"
リロケータブル逆アセンブラInside-R



10月号
特集　Game Designを考える
遊びを設計するために/ピコピコゲームが原点他
●投稿ゲーム4選
●ミュージックプログラム　ベートーベン月光
THE SOFTOUCH SPECIAL　イース/ウルティマⅣ
X68000あなたの知らない世界　BASIC to Cコンバータ
X68000BASIC入門　追撃ランダムファイル
全機種共通システム　FuzzyBASICコンパイラ拡張版
X1turbo版S-OS"SWORD"/tiny CORE WARS



11月号
特集1　全機種共通システムS-OS再考
超入門S-OS/ファイルアロケータ&ローダ
FuzzyBASICコンパイラ版BACK GAMMON
特集2　MZ-2500スペシャル　逆襲のアルゴ機能
アルゴブロック崩し/アルゴリズムを作ろう
●MZ-2500カードゲーム　KING'S COURT
THE SOFTOUCH　X68000用Kamikaze/MZ-2861用upシリーズ/トリフォニー/リバイバー他
X68000あなたの知らない世界　CP/M-68K/TITLE. SYS



Oh!X　12月号
特集　正真正銘のOh!CZ SPECIAL
新製品速報X1turboZⅡ/X1twin/X68000
X1/turboシステム&プログラミング
NEW Z-BASIC/C compiler PRO-68K
人類タコ科図鑑　第1回　Jap meets Yankee
実用(?)オブジェクト指向のゲームプログラミング第1回
●X1/turbo用カードゲームSPEED
●X68000ファイルコンバータ　MACS/HELPS
全機種共通システム　PASOPIA7版S-OS"SWORD"他

1988



1月号
特集　MZ&X拡張ボードの活用
すべての道はI/Oに通じる/MZでX1用ボードを使う
1987年度GAME OF THE YEARノミネート発表
●MZ-2500用　ALGO SPACE BLUSTER SG
●LIVE in '88　ドラゴンスピリット/悲しきチェイサー
BASICリレー連載　半熟FORTRANはいかが
X68000BASIC入門　グラフィック炎上
マシン語体操1・2・3　データ構造を考えよう
全機種共通システム　Fuzzy BASICコンパイラ奥村版



2月号
特集　グラフィック画像の冒険
X1/turboCGアニメ/トリフォニーで立体モデル
X68000グラフィックデータ/QUICK MZ PAINT他
X68000あなたの知らない世界　辞書構造/WORD POWER
マシン語体操1・2・3　Lispインタプリタ(1)
●NEW Z-BASIC詳報　その名はZ-BASIC
●LIVE in '88　グラディウス2
●SHORT ACCESS　THRILLING/POMカードポーカー
全機種共通システム　シューティングゲームELFES



3月号
特集　コンピュータサウンド"楽"入門
X1/turbo MIDIインタフェイスの製作
MZ-2500 Super Keyboard/VIPサウンドデータ公開
Oh!X LIVE SPECIAL 組曲「Ys」/Raspberry Dream他
THE SOFTOUCH Might and Magic/Hyper UD
オブジェクト指向のゲームプログラミング
X68000BASIC入門 奇襲アニメ作戦
X68000あなたの知らない世界 未公開IOCSの解析
全機種共通システム 構造型コンパイラ言語SLANG



4月号
特集　不思議の国のゲーム学
決定！　1987年度GAME OF THE YEAR
ピコピコゲーム春場所/GAME REVIEW 10本他
新製品　X68000ACE-HD/カラースキャナCZ-8NS1
X68000あなたの知らない世界　microEMACSの移植
●MZ-700　SPACE BLUSTER FX
●LIVE in '88 Moonlight Serenade/Long Night他
全機種共通システム　デバッギングツールTRADE
シミュレーションウォーゲームWALRUS



5月号
特集　BASIC入門「再検証」
BASICの歴史と意義/栄光のHuBASIC
黄金のBASIC入門プログラム/プログラミング用語集
ミュージックプログラマへの道/レイトレーシング
特別企画　言わせてくれなくちゃだワ
●新製品　X68000ACE/ACE-HD
●LIVE in '88 GET WILD/BOOM BOOM/SDI
●SHORT ACCESS 3Dボクシング/マシン語データ文生成
全機種共通システム　シューティングゲームELFES



6月号　創刊6周年記念
特集　システム環境を考える
8ビットパソコンの開発環境/Human68kのシステム環境/システムを読むためのアセンブラ入門
特別企画　究極の8ビットパソコン　8RON計画
THE SOFTOUCH　X68000用日本語ワープロEW他
●付録「あぶない福袋」
マシン語体操1・2・3　番外編　Lisp80入門
X68000BASIC入門　捨て身のミュージック
全機種共通システム　構造化言語SLANG入門　他

PENGUIN INFORMATION CORNER

ペ・ン・ギ・ン・情・報・コ・ー・ナ・ー

NEW PRODUCTS

A4サイズのノートワープロ
WV-500
シャープ



WV-500とオプション類

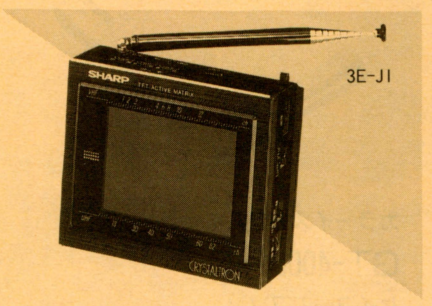
シャープがノートワープロと名づけて7月1日から発売するWV-500は，A4サイズで厚さ39.5mm，重量は電池含め1.6kgという小型・軽量のプリンタ分離タイプ。本体は138,000円，別売の熱転写プリンタWV-01TPは39,000円。

辞書は固有名詞を含め約10万語，ユーザー辞書として最大110件が登録できる。計算機能，グラフ作成機能，スケジュール管理機能，そして自動ソート/データ検索もできる住所管理機能などがある。また，ICカード用スロットを2つ内蔵しており，メモリカード（32Kバイトは8,000円，64Kバイトは12,000円）に文書登録できるほか，ゴジック体印字用カード（10,000円）もあり，8月には上海や麻雀などのゲームカードも発売の予定。また，電子手帳PA-7000の電話帳/住所録カードも同様に使える。

オプションとして，3.5インチFDD WV-10FD（40,000円）が接続でき，書院シリーズと文書の互換性が確保されるほか，パソコン通信のできる通信セットなども今後発売される。

液晶ディスプレイはガイダンスラインを含め40文字×22行が表示できる。
〈問い合わせ先〉
シャープ㈱ ☎06(621)1221，03(260)1161

3インチ液晶カラーテレビ
3E-J1
シャープ



3E-J1

240×384画素を3インチ画面に持つポケットテレビ3E-J1(57,800円）がシャープより発売された。

12メモリつきオートチューニングでVHF，UHF合わせて12局を登録でき，メモリ局と異なるものはサーチ選局で探し出せる。

また，チャンネル番号や電池交換時期の知らせなどが画面に表示される。

ビデオ入力端子つき。本体重量350g。
〈問い合わせ先〉
シャープ㈱ ☎06(621)1221，03(260)1161

電子手帳用ICカード
シティガイド東京編PA-7C8
シャープ

人気の電子手帳PA-7000用のICカードとして，シティガイド東京編カードPA-7C8（7,000円）が6月1日に発売された。

収録内容は，レストランやバー，ディスコ，百貨店／専門店，ホテル，名所など926件の首都スポット情報。また，エリア別などの条件検索機能や最寄り駅などのデータも装備している。

PA-7C8

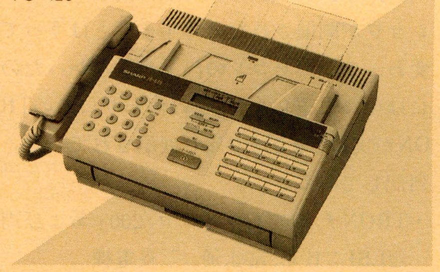


PA-7000用ICカードはほかにも多数あるので詳しくは下記へ。
〈問い合わせ先〉
シャープ㈱ ☎06(621)1221，03(260)1161

コンパクトファクシミリ
FO-420
シャープ

シャープは，5月30日にコンパクトファクシミリFO-420(298,000円）を発売した。原稿はA4サイズで10枚まで自動連続送信できる。ワンタッチダイヤル20局，短縮ダイヤル50局を装備し，相手が通信中のときは，再ダイヤルのほかに，別に登録してあるファクシミリへ転送することもできる。また，料金の安い深夜/休日に送受信するようにセットすることも可能。
〈問い合わせ先〉
シャープ㈱ ☎06(621)1221，03(260)1161

FO-420



LISP内蔵ポケコン
AI-1000
カシオ計算機

カシオ計算機が6月27日から発売するポケットコンピュータ AI-1000は，COMMON LISPの文法を参考にして厳選されたLISPを搭載したもので価格は39,800円。

32Kバイトの RAMを持ち，ファンクションキー4つ，32桁×4行表示の液晶ディスプレイを装備。数式記憶，ファイル管理はもちろん，電話番号などを記録するデータバンク機能もある。

オプションとしてBASIC，C，PROLOG，CASLなどのROMカード（各11,000円）や，

増設用32KバイトRAM (15,000円), 3.5インチFDD (49,800円), RS-232Cとセントロニクス準拠のインタフェイスを内蔵したインタフェイスボックス (16,500円) などが用意されている。

〈問い合わせ先〉
カシオ計算機㈱ ☎03(347)4811



AI-1000

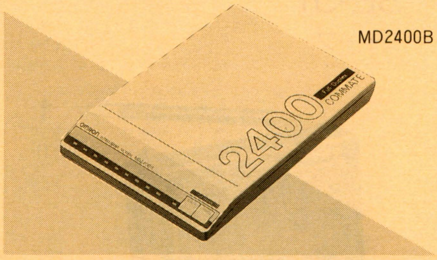
インテリジェントモデム
MD2400B/1200AII
立石電機

立石電機は, 全二重・2400bpsのMD2400B (49,800円) と全二重・1200bpsのMD1200AII (24,800円) という2機種のモデムを6月6日から発売した。

MD2400Bは, 通信エラー検出・再送方式であるMNPを搭載し, さらにフローコード最適選択機能によりコードの混乱によるエラー発生を最低に抑えたというもの。

なお, 同時に1200bpsのMD1200AIIも発売される。制御コマンドは, 両機種ともヘイズATおよびCCITT V.25bis準拠。

〈問い合わせ先〉
立石電機㈱ ☎03(436)7233



MD2400B

カラーイメージスキャナ
GT-4000
セイコーエプソン

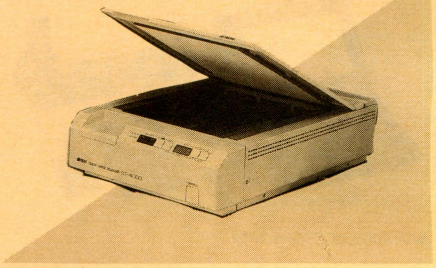
セイコーエプソンは, パソコン用カラーイメージスキャナGT-3000の上位機として, 新たにGT-4000を5月下旬から発売開始した。価格は198,000円。

GT-4000は, 1回のスキャンでA4サイズのフルカラー画像を入力する線順次走査を採用し, 読み取り時間は約90秒。読み取り画素あたり最大8ビットで入力するので, 各色最大256階調で1千6百万色の表現ができる。拡大・縮小率は50%~200%までを1%きざみで設定可能。

なお, GT-3000/3000Vをサポートしているソフトで作成したデータなどはそのまま使用できる。

〈問い合わせ先〉
セイコーエプソン㈱ ☎0266(52)3131

GT-4000



Again Watch

TRONでいっぱい

5月中旬に東京・平和島で恒例のマイコンショウが開催された。昨年までとはうって変わって, パソコンの出品が少なかったのには少々失望したが, その代わりにあのTRONが実にさまざまな形で展示されていた。列記してみると,

- TRONチップ「Gマイクロ/200」および周辺LSI：日立, 富士通, 三菱電機
- TRONチップのパネル説明：松下
- ITRON：マイクロニクス
- CTRON試作版：沖電気, 東芝
- CTRON搭載ミニコン：沖電気
- BTRON試作機：松下

といった具合。これだけTRON関係の製品が展示されたのは, もちろん初めてのことだ。さらに有料でTRON計画のリーダーである東大助教授・坂村健氏の講演まであった。これも盛況だったそうだ。

この背景には, TRON計画の推進団体, TRON協会の母体である日本電子工業振興協会がマイコンショウの主催団体であることが大きいとは思うが, ここではまず並べる製品が揃ってきたことを客観的に評価したい。とくに日立, 富士通, 三菱の3社が共同開発していた32ビットTRONチップ「Gマイクロ/200」が一般に初公開されたことはBTRONなどと違って大きな意味がある。なにしろ, TRONのオリジナルな命令セットを持つ初めての品だからだ。

TRONチップが出回り, それを使ったパソコンが製品化され, ここでOSとしてBTRONやITRONが走り出すにはまだしばらくは時間がかかりそうだが, そろそろTRONコンピュータの輪郭が見えてきたといっていいのかもしれない。

4Mビットメモリが離陸

これもマイコンショウからの話題。マイコンではなく, メモリチップも今回のマイコンショウでは話題を集めた。一部で予想されていたことだが, 4MビットのDRAMの試作品が公開された。展示したのは東芝, 三菱電機, 日立製作所の3社。富士通と松下もスペックをパネル展示した。

詳しいスペックはここでは省かせていただくが, おおむね各社の公表データは同じで, プロセスルールは1Mチップより20%細い0.8μルール。アクセス速度は100ナノ秒前後。チップサイズは1メガや256キロチップと同じ。

4M時代が目前であるとは昨年からいわれてきた。しかしこうやって, 実際に目の前に試作品を置かれると, いよいよか, という感じが実感としてわいてくる。

実はこの4MDRAM, 単純にメモリは大きいほうがいい, といった問題ではなく, 深刻な需要がすでにある。新しいパソコン用OSであるOS/2がその引き金なのだ。OS/2を使う場合, メインメモリは最低でも3~4Mバイト必要であることは以前説明したと思う。たとえば4Mバイトを装備するとしよう。こうなると, 256KビットのダイナミックRAM (DRAM) の場合, 実に128個が必要になる。こうなるとメモリボードだけでパソコンの本体の中がいっぱいになってしまいそうなので, 事実上, 使用に耐えない。だから最低でも1MビットのDRAMが必要となるわけだ。しかし1MビットDRAMでも32個必要だ。これでもかなりのスペースを食ってしまう。その点, 4MビットのDRAMがあれば8個ですむから理想的だ。

さてマイコンショウが終わって, 4MビットのDRAMがいつから商品化されるかを気

インテリジェントリモコン
R-65
ビッグサンズ



R-65

AV機器用のインテリジェントリモコンの新製品R-65 (12,500円) がビッグサンズより6月1日に発売された。キー数は65で，どのキーも登録内容は変更可能。3 個のLEDランプが，送信/受信/エラーの状態を表示する。

単3電池4本使用，サイズは幅200×長さ110×厚さ18mm。

〈問い合わせ先〉
ビッグサンズ㈱ ☎06(311)0077

INFORMATION

今夜も朝までPOWERFULまあじゃん
ギャルコンテスト
デービーソフト

ギャルコンテストで好きな女の子を選んでプレゼントをもらおう。「今夜も朝までPOWERFULまあじゃん」のエキサイト麻雀モードで登場する7人の女の子から，一番気に入った子を選んで，パソコンショップ備え付けのハガキ（官製ハガキも可）に記入し，下の宛先に送付する。1位になった彼女に投票した人の中から，抽選で500名に彼女のオリジナルデータディスクをプレゼント（ただしこれは「今夜も朝までPOWERFULまあじゃん」のディスクがないと使えない）。応募の際は，住所・氏名・年齢・希望の機種メディアを明記すること。宛先は，
〒060 北海道札幌市中央区北1条西7丁目
住友海上札幌ビル デービーソフト株式会社
応募期間は6月末日まで。

〈問い合わせ先〉
デービーソフト㈱ ☎011(251)7462

BOOK

X68000ガイドブック
ビジネス・アスキー

X68000活用本の類の最新刊。「生いたち」や商品コンセプトなどのプロフィールに始まって，システム構成，標準ソフトなどをマニュアルに即して解説している。

『X68000ガイドブック』 佐藤日出男著
B5判，295ページ，2,800円
〈問い合わせ先〉
㈱ビジネス・アスキー ☎03(486)7119

X68000
ガイドブック



にしていたところ，5月下旬になって，新聞に続々と，4MビットのDRAMの試作品が提供され出したというニュースが載り始めた。いろいろ記事が出たのだが，総合すると5月までに試作品を出荷したのが日立製作所，富士通，三菱電機の3社。6月には東芝が出し，7月には日本電気が続く，という。また松下やテキサス・インスツルメンツ社も計画中とか。

記事によれば，試作品ではなくて本当の製品が販売されるようになるのは来年後半以降だろうという。そのスケジュールから考えれば，パソコンで気軽に使われるようになるには早くても2年後ということになろう。2年後になれば，OS/2のアプリケーションソフトも出揃っているだろうから，ちょうど都合がいい。2年後。PC-9801はそのときでも独占状態にあるのだろうか？ とにかく4Mチップ時代に向かってテイクオフした。

AXマシンが動き出す

5月はにぎやかだった。マイコンショウの翌週，今度は東京・晴海でこれまた恒例のビジネスショウが開催された。

こちらはマイコンショウとは違って，現在販売中の商品と近く発売されそうな製品のオンパレード。チップではなく完成品ばかりだ。

4日間で実に46万6千人もが詰めかけたそうで，混雑を極めていたが，見どころはタップリあった。最新型ワークステーション，ラップトップパソコン，パソコン新製品，ワープロ。ISDN，パソコン通信，LAN。オフィス家具。電子文具。果てはシステム手帳ブームに乗って，文房具コーナーもなかなか人気の的。デモンストレーションにも力が入っていた。

どれひとつを取りあげても20行以上コメントを書けるのだが，ここでは一番注目を浴びていたうちのひとつについて紹介しよう。

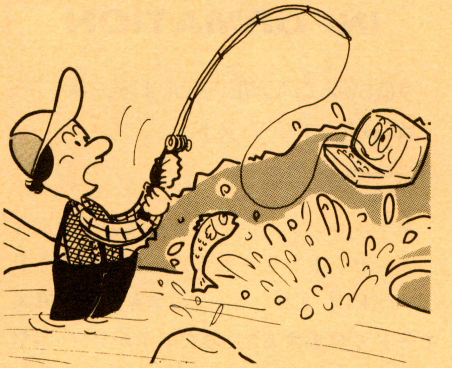
パソコン新製品ではついに話題のAXマシンが大挙して姿を見せた。すでに販売活動に乗り出している三洋電機とシャープ，三菱の3社が，これから発売するAXパソコンを一斉に公開したのだ。三洋は卓上型に加えてラップトップも用意。三菱はラップトップとその改造機の疑似卓上機を出品。そしてシャープは80386をCPUに使い，さらにGSPを装備して高速画像処理を実現した超豪華マシンを投入した。三洋はすでに販売している80286の卓上機だけでなく，32ビット高速機とラップトップ機も追加，展示した。

いよいよAXが姿を現した。会場でも目立って人気を集めていた。

「もうPS/2が定着したからAXなんて……」という声もチラホラと出ているが，実際にはそんなことはない。米国ではIBM-PC/AT互換機が売れ行きアップを示しており，日本でもAT互換機である東芝のJ-3100が依然として好調だ。AT互換機市場はまだまだ健在である，といえよう。そんなおりに彗星のようにデビューしたAXマシン。そこそこは行けそうだ。今後の各社製品の売れ行きについては，後日このコーナーで報告しよう。

なお，シャープのAXパソコンは，通称「AX386」と命名されたが，型番は「MZ-8702/6」という。シャープはMZという名前に特別な価値や重い歴史を感じられなくなったようだ。MZの歴史にピリオドを打った展示会だったのかもしれない。そう思うのは短慮にすぎるだろうか。 (K.T.)

STUDIO X ON AIR



FROM READERS TO THE EDITOR

すっかり初夏の気候となり，そろそろ南のほうから海開きの話題などが聞こえてきそうです。皆さんはこの夏，どんな予定を立てているんでしょうか。そのような話も，機会があればぜひ聞かせていただきたいですね。

◆5月号の「BASIC特集」はたいへんよかったです。それに「言わせてくれなくちゃだワ」も，読むのはキツかったけど面白かったです。なかでも最も驚いたのは85ページの機種別保有者数で，X68000ユーザーがこんなに多かったことにはたいへんうらやましく思えました。

燈田　安幸（15）大阪府

◆「BASIC特集」でHuBASICがほめられていたが，いいところばかり書いてあるのは問題だと思う。X1の売れ行きが伸び悩んだのは，私はあのペイントルーチン＝ハドソンのせいだと思っている。

山本　正幸（18）神奈川県

山本君のいうようにいろいろと不満はあったかもしれないけど，あのころのHuBASICって，結構いい線いってたと思うんですけどね。でも5月号の特集を読んで，多くの方から「BASICを見直すきっかけを得た」といった内容のご意見をたくさんいただいて，編集室でも嬉んでいます。

◆僕は去年の9月号からOh！X（Oh！MZ）を読み始めたので，「言わせてくれなくちゃだワ」は今回が1回目でした。読んだあとの感想は「読者パワーを思い知らされたっ！」のひと言に尽きます。それにしても凄いですねぇー。

北野　卓哉（15）大阪府

◆もしかして5月号96ページ左下の「鳥居勉さん」は，栃木県ということからしても“Xファミリーの生みの親”である，シャープ電子機器事業部長の鳥居さんではないだろうか。第1回の祝一平氏といい，Oh！Xはとんでもない隠れキャラを用意するのが好きですね。

田中　義彦（24）東京都

◆お詫びするのである。まさか私のしょーもないメッセージが，第3回「言わせてくれなくちゃだワ」に載る（87ページの最後のあたり）なんて夢にも思っていなかったので，ついつい，いいかげんな嫁のグチなんぞ書いてしまったが，うちの女房はたいへんよくできた嫁である。よって5月号の私のハガキには，一部“ウソ”があったことをここに深くお詫びするのである（頭ペコー）。

田中　伸和（30）大阪府

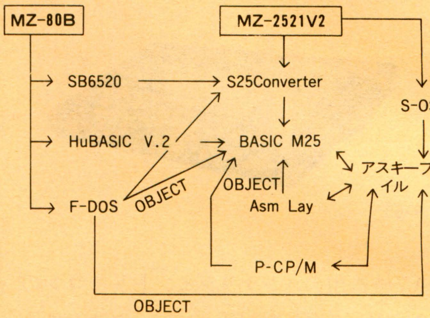
これはもう，ハガキの一部にウソを書いてしまったお詫びというより，田中さんから奥様への体面繕いと見えますが，いかがなものでしょうか。

◆5月号86ページの伏喜さんの上のCGを見てなにか思い出しませんか。そーです，あの有名なデービーソフトの「南太平洋アドベンチャー」です。あのゲームこそこの私が初めて買ったゲームソフトだったのです。あのゲームって，怪物にいきなり挨拶するとか，オールで戦うとかまったくむちゃくちゃなものでした。はっきりいってアタッチなんぞよりも超卑劣なものであったといまでも確信しています。う゛ーん，懐かしい。でもやっぱりアイスクリームは雪印のパナモです。

稲野辺　弘（17）神奈川県

たくさん届いた5月号のハガキのなかに，「雪印のパナモは美味しい」と書いてきた人が稲野辺君のほかに3名ほどいたのですが，いったい“パナモ”なるアイスクリームのどこがウケているんでしょうね。

◆「言わせて～」の「私が主役だぁ」のトリに載せていただきどうもありがとう。掲載されるのを狙ってハガキを出したのはあれが初めてです。さて，これからはなにを書いて出しましょうか。それと，ようやく私の手持ちの各システムにおける交流が図れる環境が整いました（といっても80Bだけは一方通行ですが）。



ここまでひとりで仕上げるのにはちょっと時間がかかりましたが，これもすべてOh！X（Oh！MZ）のおかげです。

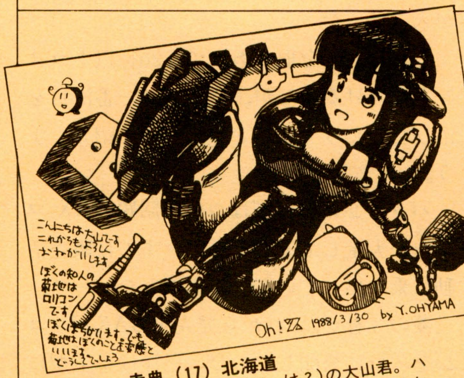
棚瀬　小三郎（61）北海道

いえいえ，これらのシステム環境を作られたのはひとえに棚瀬さんの努力の賜でしょう。これからもその環境を生かしてがんばってください。

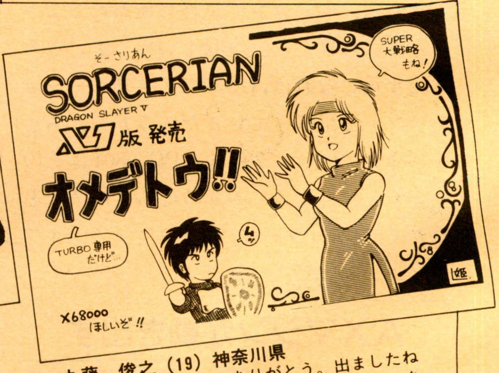
◆わ，悪かった。吉田幸一さんがそこまで考えているとは思わなかった。僕は1～7巻までの半分しか読んでいなかったので，そこまで意見はできない。でも，結局は人それぞれなんですよね。

塚田　弘（18）栃木県

◆X68000ACE-HDでは，またまた新しいカスタムICを作りまくったようですね。それでもってその名前が“メシア”だって？　いったいブッタはどこへ行ってしまったんでしょうかね。やはり仏様より救世主のほうが凄いんでしょうかね。もし，今度また新しいICを作るようなことがあったら名前はもう“ジーザス・クライスト”しかないんじゃないでしょうか。そうそう，ついでに漢字ROM & IOCSには“金田一春彦”とか，スプライトコントローラには“ピカソ”なんていうのもいいんじゃないでしょうか。ついでに数値演算プロセッサボードには“広中平祐”とかにしちゃいましょ。最後に初代X68000用20MバイトHDは“アンドレ・ザ・ジャイアント”なんて呼ぶようになったら，世の中楽しいでし



▲大山　幸典（17）北海道
トップバッターは初登場（だっけ？）の大山君。ハガキからはみ出しそうなダイナミックな絵ですね。うーん，今月はなかなかレベルが高いぞ。



▲丸藤　俊之（19）神奈川県
丁寧なイラスト，どうもありがとう。出ましたね え，ソーサリアン。4MHzでも，ハードウェアスクロールがなくても，X1turboは速いのだ。

ょうね。　　　岡部　玄（22）長野県

スプライトコントローラには“山形導博”，数値演算プロセッサには“玲於奈”なんていうのもいいかもね。

◆こんなに街の緑が鮮やかなのに，僕の心のなかは真冬になってしまいました。あまりに突然だったから，どうすることもできなくて……。いままでずっと一緒だったから気づかなかったけど，たぶん僕も甘えていたんだと思います。これぱかりは誰を責めることもできないし，誰も悪くはないんです。きっとそうに決まっています。少し時間はかかるかもしれないけど，また，あのときみたいに2人で海を見に行けたら嬉しいなって，ただそれだけです。“いつだってひとりより2人のほうがいい”，ほんとうにあの歌のとおりだと思います。だから……。あれっ，これどこ行きのハガキでしたっけ。

井上　武司（18）神奈川県

このテのハガキはタイムスリップでもして，レモンちゃんのセイヤング，またはアンコウさんのオールナイト，ナマズのおばさんの走れ歌謡曲，またはナッチャコパック（テーマによっては可）にでも送ってみれば，きっと番組の最後のほうで読んでもらえるはずです。あっ，失礼，井上さんはまだ生まれてなかったのかな，あのころって。

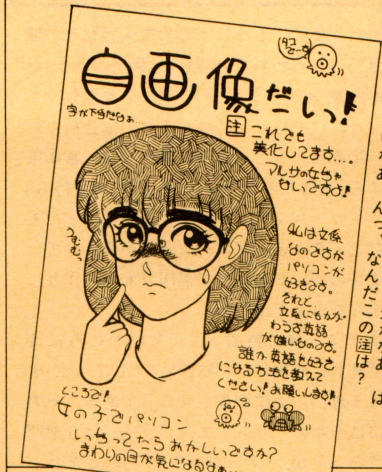
◆最近，パソコンのことをバカにしていた妻が，「スーパーレイドック」をやり始めてから病みつきになってしまい，9カ月の娘を私に押し付けて，毎日，毎日パソコンの前に座りきりになっている。私がパソコンにさわっている時間がなくなってしまった。いったい，どうしてくれるんだ。　　　深沢　芳明（30）神奈川県

◆あと1週間で「ソーサリアン」である。日本全国「ソーサリアン」である。編集室にサンプル版はあるのか「ソーサリアン」である。木屋氏を人間国宝にしたい「ソーサリアン」である。古今東西「ソーサリアン」である　（編集室注：以下似たようなのが延々と続くので一部省略の「ソーサリアン」である）。ああ，「ソーサリアン」，「ソーサリアン」，「ソーサリアン」なのである。　　　上杉　直久（18）神奈川県

いやー，上杉さんのほかにも「ソーサリアン発売」のニュースに関してはずいぶんとご意見をいただきました。turbo 専用なのが悔いが残りますが，仕上がりは今月のレビューでお届けしたとおりの結果です。6月24日発売の「イースII」も期待大ですね。

◆突然ですが，X68000版レリクスの責任者は実際に尼寺に行ったのでしょうか。もし行ったのなら，社名が「ボーズテック」になってたりして。チャンチャン。　龍尾　謙二（21）岐阜県

◆私の X68000は両親の上海専用ゲームマシンと化している。毎日20～30回もやっていて，父母ともどもギャーギャーいいながらマウスをクリックしています。それでも毎日1度はクリアする上達ぶりには息子の私もただ感心するばかり。また，最近では源平にまで手を出しており，バアさんの声に大爆笑しています。



◀小林　聖美（？）青森県
やった～，今月も女性からの投稿だ。しかも自画像。年はなんぼかなあ。女子大生かなあ。はやく逢いたいなあ。んっ，なんだこの㊟は？



◀大村　和弘（？）愛知県
おおっ，なんとアドル様でいらっしゃいましたか。これはまあ，立派になられて，じいはうれしゅうございますぞ（一瞬，また自画像かと思った）。

苗木　純生（21）大阪府

◆ X68000の源平討魔伝って凄いですね。しゃべることしゃべること。僕は後ろに人が隠れているんじゃないかと思っていました。

信太　徹（18）高知県

ホント，最近のゲームはよくしゃべりますね。X1版の「ゼリアード」や「ねぎ麻雀」もしゃべりまくっています。でも夜中にひとりでやってると，ちょっと不気味な気もしないではないのですが……。

◆夢のサッカーやサイエンスロマンキャンペーンといった催し物で，快進撃を続けている我らがシャープだが，ここ静岡でもシャープ製品10万円以上お買い上げで，もれなく「石川さゆりショウご招待」である。

竹谷　直樹（19）静岡県

夢のサッカーより，もれなく「石川さゆりショウ」のほうがシュールでいいなぁ。こういうセンス大好きです。

◆今年めでたく国立大学に入学できて，喜びを体いっぱいで感じている今日このごろです。もっと嬉しいことにうちの学校の購買部にはX68000が置いてあり，学生が好き勝手に使えるのです（その隣にはPC-98もあったりする）。やったぁー，これでX68000を買わなくてもスペハリや源平ができる。この学校に受かってほんとによかった。　　　小松　元弘（18）静岡県

こんな話，シャープさんが聞けばきっと泣いちゃうだろうな。でも小松君も同じ静岡なんだから，いま X68000を買えば「石川さゆりショウ」ご招待だよ。

◆先々月まで私は FM ユーザーだった。しかし，友人が X1turbo からエプソンに乗り換えたのを機に，この私も以前から憧れの X68000を買ってしまった。さあ，もうためらうことはありません。どんどんソフトを出しなさい。私はいいものはたくさん買う社会人ユーザーだぞ！

辻口　恭明（30）北海道

ずいぶんと威勢のいい方が仲間に加わっていただいて，頼もしい限りです。それに推薦するソフトのところに堂々と「源平討魔伝，X68000を買ってよかった」と書いてくれた辻口さんの性格はもっと好きです。

◆ついにシャープが重い腰を「どっこいしょ」と上げたようですね。皆さんはもう気づいていると思いますが，なんと X68000の広告を電車の中吊り広告のなかに見つけたのです。あれは見る人に大エジプト展を思わせるような，黄金の輝きを持った感動を与えます。この広告はいままでと違ったシャープの意気込みが感じられてとても Good です（マル）。

石部　忠之（19）千葉県

◆いやー，やっとレイトレーシングの記事が出ましたね。なんせ過去4年ほどレイトレーシングの記事があったという記憶がありませんでしたからね。ところで，ふと気づいたのですが，Oh！Xには「創刊○周年記念」と「改題○周年記念」または「Oh！X独立○周年記念」の2大行事があるわけですが，やはり両方ともちゃんとなにかやるんでしょうか。私はこうなった以上「言わせてくれなくちゃだワ○周年記念」と「S-OS 発足○周年記念」を追加して Oh！X 4大行事まで発展すると読んでいるのですが。

宮本　康司（19）兵庫県

そうですねえ，創刊と改題と両方やれば1年に2度お祭り騒ぎができていいんでしょうけど，そのうち「年中お祭りやってるアッパレマガジン」なんてサブタイトル付けられそうで怖いなぁ。

◆あのですね，「プログラムを入力する」とよく



◀田村　憲生（19）広島県
げっ，ウマイと思ったら呼び捨て御免の田村ではないか。なに，呼び捨てはイヤ？　え，ノリちゃんと呼んでくれだって？　いいのか本当に……。

いいますが，なぜ「入力」というのでしょうかね。「入力」という文字を単純に分析すると，力を入れるという意味になるわけで，これでは意味不明ですね。誰が最初に「入力」といったのか，誰か知っていたら教えてください。

荒木　光弘（32）東京都

入力というのは「INPUT」の訳語なんですが，英語のINPUTはエネルギーや信号を機械に供給するという意味なので，そうなっちゃったわけですねえ。

◆さて，ここでクイズです。次のうちいちばん不気味なコンサートはどれでしょう。1）最後に全員で涙を流しながら「ネコの森には帰れない」を合唱する谷山浩子のコンサート。2）同じく「防人の歌」を合唱するさだまさしのコンサート。3）同じく「秘密の花園」を合唱する聖飢魔IIのコンサート。

畦地　真太郎（18）北海道

はっきりいって，全部不気味です。

◆山形大学工学部情報処理系に受かったMARO君は，ウキウキしながら下宿生活を始めたが，彼を悲しませることがあった。次のうちから最も彼が悲しんだと思える番号を選んでください。1）民放が2局しかない（彼はアニメファンだった。）2）父親に「テレビがほしい」といったら，3インチ白黒テレビを渡された。3）テレビに色がついてなかった見返りにCDラジカセを買わそうとしてショップに行くと，彼のほしがっていた品物は3週間しないと入荷しないといわれた。4）仕方がないのでMZ-2000を持って行こうとしたら。ディスプレイとキーボードがダメになっていた。5）これじゃあんまりなので友だちにカシオのポケコンを借りたが，代わりにガラカメ34冊を貸すハメになった。6）バイトしてパソコンを買おうとしたら，親に「バイトなんかしないで4年で卒業してくれ」攻撃にあった。以上ゲゲボクイズ風にまとめてみました。

佐藤　啓之（20）山形県

当然，MARO君って佐藤さんのことなんでしょ。それにしてもなんでも親に買ってもらおうというその根性は凄い。

◆私の友人が東京の大学に行くことになったが，彼は自分のX1turboやワープロは持って行かなかった。そこでもうひとりの友人と私との間で形見分けが行われた。協議の結果，私のX1turboはFM音源，イメージボード，プリンタが付くこととなった。やはり「持つべきものは金持ちの友人」である。

黒田　信和（18）富山県

この「金持ちの友人」のほかに，「大金持ち」や「ガールフレンドが多い」など，まだまだたくさんのバリエーションが残されていますので，がんばって黒田君も友人作りに励んでみてください。

◆アカデミー賞の特徴（長でないところに注目），「売れた映画はいい映画」，そして「いい映画には5部門以上に授賞させる」である。その証拠には今年の「ラストエンペラー」は9部門授賞であったが，「アンタッチャブル」がそれよりもっと素晴しい映画であったことも事実である。考えてみると，「アンタッチャブル」が「ラストエンペラー」に劣っていたところは，興行成績を除けばエキストラの人数と製作費くらいではないかと思う。それにしても1本の映画に9部門も授賞させるのはやめてほしかった。

陣山　達矢（18）大阪府

そうですね，陣山君のいうようにラストエンペラーは東京で未だに上映されているところを見ると，確かに「売れた映画はいい映画」というのは当たっているような気がします。

◆コンピュータからかけ離れた話で申し訳ないのですが，『なんてったってウルトラマン』という本はなかなか面白いですよ。このテの本はほとんどが小学生向けで，意味のない情報（ウルトラマンの身長とか必殺技など）しか得られないものですが，大人の目から見たウルトラマンというか，ウルトラマンが誕生するまでの企画段階の話や，円谷プロやテレビ局の考え方とか実相寺昭雄監督によるウルトラマンの話など載っています。要するに少年時代にウルトラマンとその仲間を見て育った我々20歳代の人が，もう一度ウルトラマンの世界を見直すのにはうってつけの本といえそうです。まあ，880円くらい（友だちに本を貸してしまったから正確な値段を忘れてしまった）と，安いか高いかわからないような値段なのでお勧めです。

堀　僚嗣（22）大阪府

堀さん知ってます？　東京地方ではなんと，朝の4時30分から強力なウルトラマンシリーズ2本立てが，いまもテレビで放映されているんですよ。

◆グァン!!　ガタガタガタン。ただいま午後11時50分30秒。桜島からの噴火の衝撃波が部屋に届きました。編集室に部屋の窓の外でとれた火山灰を送ります。

塚本　雅俊（19）鹿児島県

わざわざセロテープに桜島の灰をくっつけてハガキで送ってくれてありがとう。みんなで触って遊んでます。

◆アパートでのひとり暮らしを始めて約1カ月。その間いろいろなことがありました。いきなりX1専用ディスプレイが煙を出して故障し，その修理代に1万5千円もかかってしまった。せっかく貯めたお金でマウスを買おうと思っていたのに……。人生ってこんなものでしょうか。それから，5月号では定期購読の案内をありがとうございました。あれって僕が前にハガキに書いてお願いしたからなのかな。

飯野　直樹（18）神奈川県

定期購読の件については，飯野君のほかにもたくさんの方からおハガキをいただいたので，遅くなりましたけどご紹介させていただきました。

◆私は埼玉に住んでいますが，その埼玉で「'88さいたま博覧会」などというものが開催されており，先日，私も友人とともに行ってきました。さて会場はさまざまな屋根を持ったパビリオンがたくさん並び，あの「つくば博」を思い起こさせます。まず手始めに東芝館へと向かいました。待ち時間はかなりのものでしたが，つくば博に比べればかわいいものです。なかでは3DとCCDを使ったラジコンカーで遊んできました。そのあといくつか見て回りましたが，3D映像をウリにしている館が多いようです。HSSTにも乗ってきましたが。走行距離が短く，乗り心地といわれてもピンとこないのが実感でした。

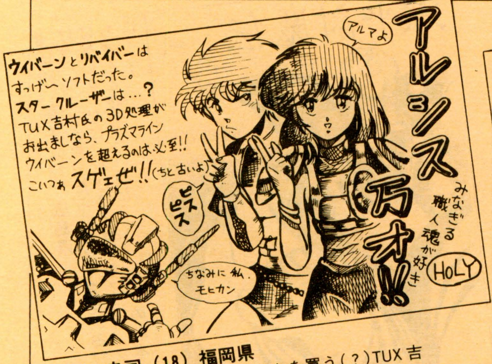
白石　孝（18）埼玉県

いまは地方博のオンパレードみたいですね。でもウリものがHSSTだったり，パンダだったりというのは，どうも地方博らしさがなくって個性に欠けるような気がするんですけどね。

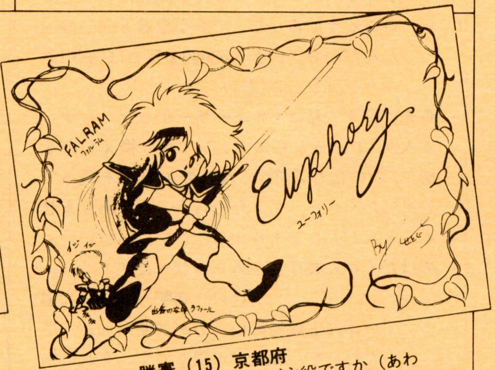
◆私は春休みに北海道に行ってきました。そのときに青函トンネルを通りました。これからこのトンネルを利用される方のためにお教えしておきますが，本物に突入するまえにフェイントの小さなトンネルがありますので，そこではしゃいでハジをかかないようにご注意を。それから，襟裳岬のJRバス待合室（燈台ロ）の見える場所に置いてある近鉄の切符は，この私のものです。

坊農　誠（17）奈良県

青函トンネルっていうのも近いうちにぜひ一度は行ってみたいものですね。でもフェイントのトンネルっていったいどちら側にあるんでしょうか。ハジをかかないためにも今度また教えてくださいね。



▲堀　幸司（18）福岡県
そりゃあもう，泣く子もソフトを買う（？）TUX吉村氏の3Dグラフィックですから，期待しますよね。ちなみに，アルマちゃんお元気かしら。



▲瀬戸口　勝憲（15）京都府
フフフ，やっぱりファルラムが主役ですか（あわれなラファール）。でもユーフォリーってハガキ見てると結構評判いいみたいですね。

ぼくらの掲示板

●掲載ご希望の方は，官製ハガキに項目(売る・買う・氏名・年齢・連絡方法……)を明記してお申し込みください。
●ソフトの売買，交換については，いっさい掲載できません。
●取り引きについては当編集室では責任を負いかねます。
●応募者多数の場合，掲載できない場合もあります。

仲　間

★「FSCM」ではMZ-700/1500ユーザーを対象とした会員を募集します。活動は主に会報を中心とした情報交換などですが，今後，会員の皆さんからの意見もどしどし取り入れていきたいと思っています。また，ほかのMZ-700/1500ユーザーズクラブの皆さんとの交流を図りたいと考えていますので，各ユーザーズクラブの代表者の方からのご連絡も併せてお待ちしています。　〒485　愛知県小牧市南外山北官舎C-2-103　前田純之介

★MZシリーズを対象としたサークルを今度発足させようと思っています。まだサークル名などは決まっていませんが，興味のある方は機種名，パソコン歴，活動内容に関するご意見などを添えて，連絡先を明記のうえ封書にて連絡を。　〒203　東京都東久留米市下里7-8-13-502　野口貴史　(20)

★「MXC」ではX1ユーザーの会員を募集します。主な活動内容はゲームの情報交換や月1回の会報発行などです。ゲームファン，テープユーザー，初心者大歓迎。詳しいことは60円切手同封のうえ封書にてご連絡を。　〒992　山形県米沢市東町1-1-50山口アパート206　水口昌郁　(19)

★X1/X68000ユーザーの皆さんを対象としたXユーザーズクラブ「WATER」では会員を募集します。当クラブでは初心者からその筋の方まで幅広く楽しんでいただける会報作りをしています。そのほかにも太田貴子ファンの方や小幡洋子ファンの方も大歓迎。興味のある方は60円切手同封のうえ封書にて連絡を。　〒975　福島県原町市小川町109-2　山崎潤一　(19)

★「WC」ではX1およびPC-88ユーザーを対象とした会員を募集します。活動は主に同人誌やオリジナルソフトの共同製作を中心に行っています。会報は毎回みんなが楽しめるものに仕上げようと努力しています。興味のある方は60円切手同封のうえ封書にて連絡を。　〒734　広島県広島市南区仁保3-40-28　津田敏之　(18)

★「YUME-NET」ではX1シリーズ，FM-7シリーズ，PC-6000シリーズを対象とした会員を募集します。活動は会報を中心にFORTRAN, COBOL, BASICの講座，プログラムコンテスト，ゲーム・ビデオ情報などを行っています。入会ご希望の方は60円切手同封のうえ封書にて連絡を。　〒849-23　佐賀県杵島郡山内町大字宮野26167-1　森伸二　(17)

★「SPC」では第3期会員を募集します。対象はS-OSユーザー，シャープ系パソコンユーザーであればどなたでも結構です。特に音楽に興味のある方，ハードに強い方など大歓迎。また，同時に交流サークルも募集しています。詳しくは60円切手同封のうえ封書にて連絡を。　〒036　青森県弘前市東城北1-6-3　菊池淳　(17)

★「倶楽部FIGHTING」はX68000ユーザーからファミコン，ナイコンまで幅広く集まったゲーム中心のサークルです。活動はゲーム，マンガ，ビデオ，小説，音楽など豊富な内容の会報を中心に行っています。会費は会報代として330円のみ。詳しいことは60円切手同封のうえ封書にて連絡を。　〒701-42　岡山県邑久郡邑久町尻海2738　元山寛　(17)

売ります

★MZ-1500用RAMファイルMZ-1R18，ボイスボードMZ-1M08，データレコーダMZ-1T03を各5千円で。連絡は往復ハガキで。　〒948　新潟県十日町市本町6-2　高橋守　(17)

★MZ-2500用テレシステムズ製の640KバイトRAM DISKの未使用新品を1万5千円で。または増設用VRAMか辞書ROMとの交換も可。連絡は往復ハガキで。　〒001　北海道札幌市北区屯田5-8-4-12　後藤修　(17)

★プロッタプリンタCZ-8PP2(1年間使用)，漢字ROM内蔵を1万5千円で。漢字プリンタTR-24(2年半使用)を2万円で。いずれも送料込み。連絡は往復ハガキで。　〒952-13　新潟県佐渡郡佐和田町中原313-1青柳寺団地146　石塚孝之　(31)

★X1用プリンタCZ-8PC2を3万5千円前後で。連絡は往復ハガキで。　〒675　兵庫県加古川市野口町野口542-23　藤原顕治　(17)

★X1用データレコーダCZ-8RL1マニュアル付き完動品を1万円で。できれば近郊の方に手渡し希望。連絡は往復ハガキで。　〒167　東京都杉並区今川1-1-1今川荘　村井裕弥　(30)

★X1用データレコーダCZ-8RL1の新品同様(3日間だけ使用)を送料別1万円で。付属品付き。またはCZ-8BE2との交換も可。連絡は往復ハガキで。　〒520　滋賀県大津市蓮池町13-19　福島義浩　(19)

★X1用カラーイメージボードCZ-8BV1を1万円前後で。連絡は往復ハガキで。　〒020-01　岩手県盛岡市みたけ4-11-56　佐々木誠徳　(18)

★X1用増設FDD・CZ-503F×2基を4万5千円で(ただしI/Fは1枚のみ)。バラ売りの場合はI/F付き2万5千円，FDDのみ2万円で。できれば2基セットで買ってくれる方希望。また，X1用I/OポートCZ-8EPを5千円で。連絡は往復ハガキで。　〒849-11　佐賀県杵島郡白石町大字福吉2003-2　石橋和史　(17)

★X1用プリンタCZ-8PD2の付属品，箱付きを2万円前後で。またX1turbo専用ディスプレイCZ-855DBの付属品，箱付きを5万円前後で。どちらも1年半ほど使用。連絡は往復ハガキで。　〒478　愛知県知多市つつじが岡4-13-2　内藤陽一　(21)

買います

★MZ-700用データレコーダMZ-1T01を4千円で。完動品であれば傷などは可。連絡は往復ハガキで。　〒624　京都府舞鶴市宇伊佐津327-23　田中智樹　(16)

★MZ-1500用RAMファイルMZ-1R18を送料込み1万円前後で。連絡は往復ハガキで。　〒013-01　秋田県平鹿郡平鹿町浅舞字浅舞136　照井清和　(17)

★MZ-2000用ユニバーサルI/OカードMZ-8BI01の完動品を送料込み2万円前後で。連絡は往復ハガキで。　〒497　愛知県海部郡蟹江町百保27-2　山森克己　(43)

★MZ-2000用FDDインタフェイスを1万円で。連絡は60円切手同封のうえ封書にて。　〒065　北海道札幌市東区北23条東12-1　三上義広　(19)

★X1用カラーディスプレイCZ-600DまたはCZ-880Dを6万～7万円の範囲内で。またチルトスタンドを安価で。300/1200ボーのモデムを1万～1万5千円で。連絡は往復ハガキで。　〒596　大阪府岸和田市磯ノ上町4-10-7　加藤哲男

★X1用熱転写プリンタCZ-8PC2(I/Fを含む)，またはCZ-8PC1＋第2水準漢字ROM・CZ-8PC1-3をセットで3万8千円で。連絡は往復ハガキで。　〒344-01　埼玉県北葛飾郡庄和町米島366-4　松本伸行　(16)

★X1用FM音源ボードCZ-8BS1(付属品すべて込み)を送料込み1万～1万5千円で。連絡は往復ハガキで。　〒977　福島県田村郡三春町会下谷18-10　石田実　(18)

★X1用FDDインタフェイスCZ-8BF1＋ディスクBASICを5千～1万円くらいで。　〒790　愛媛県松山市久米窪田町443久米住宅233　尾埜繁　(33)

バックナンバー

★Oh！MZ1986年1，2月号を送料込み各1,000円で。切り抜き不可。連絡は往復ハガキで。　〒105　秋田県本荘市出戸町小人町125-3　畠山拓真　(15)

★Oh！MZ1982年6，7，8月号，1984年・1985年の各1～12月号，1986年の1～5，8，9月号を1,000円で(1984年の各号については1,200円で)。多少の汚れ可，切り抜き不可。連絡は往復ハガキで。　〒815　福岡県福岡市南区大橋3-15-43　大橋新光ハイム410号　兼重宣康

編集室から from E・D・I・T・O・R

DRIVE ON

このコーナーでは，本誌年間モニタの方々のご意見を紹介しています。今月は5月号の記事に関するレポートです。

●「手段としてのBASIC」に同感です。私自身，BASICのプログラムを作っていて，できなかったことは何ひとつありません（スピードが遅いだけです）。すべてBASICでできました。RS-232Cを使って温度モニタプログラムも作りましたよ。HuBASICのユーザー寄りの構成は，本当に使いやすいと思います。

佐藤　孝（31）MZ-2500　埼玉県

●他機種，とくにMZ-80K/Cの場合はHuBASICが最高とも思えるが，我がMZ-1500になるといちがいにそうとは言えない。BASICが巨大になったため漢字処理機能がなくなった。あのHuBASICなのだから，S-BASICでサポートされていない辞書ROMのサポートくらいしてもらえるかと考えたがダメだった。加えて実数版はおろか，整数型コンパイラも出なかった。そしてあの値段である。確かに1万円は言語ソフトのなかでは安いのかもしれないが，ぜひコンパイラもおまけでつけてほしかったと，1ユーザーは言いたい。

関根　孝司(19)MZ-1500, PC-1480U　東京都

●HuBASICほどマシンに密着したBASICはないと思う。マシン単体をみてもX1はとんでもなく多機能だが，それをオプション機器を除いてすべてサポートしたHuBASICはすごい。巨大なBASICゆえにフリーエリアが小さい，スピードがちょっと，という難はあるが，ユーザーからみたBASICで，これ以上のものはないと思う。また，個人的には「美しくないからBASIC」なのだとも思っている。字下げや桁揃えなどをして見やすいプログラミングを心がけてはいるが，スパゲティプログラムでもBASICだから動く，というところにそこはかとない魅力を感じているのも事実だ。

山口　幸一(22)X1turboII, JR-100, PC-E200　宮崎県

●5月号の特集を読んで反省しました。BASICで多くのプログラムを組んできたにもかかわらず，少なからずBASICを馬鹿にしているところもあったのです。しかし，確かにこれほど手軽にプログラミングできる言語は珍しい。最初のBASICがたった14のコマンドしか持っていなかった，ということにも改めて感慨をもよおしました。グラフィック，ミュージック，それにディスク入出力関係などを除けば，ほとんどのプログラムはこの14コマンドで書けてしまうのでしょうね。そして，X1が8ビットの世界で長く活躍できるのは，開発コンセプトの良さとHuBASICの存在によるところが大きいと思います。

薬師神　昌夫（17）X1turbo Z　愛媛県

●「結局，最後は腕力が勝負です」という，特集の「美しいBASICの学び方」の考えに賛成です。ひらめきみたいなものも重要ですが，それは経験を積めば誰もが得られるものでもあります。つまるところ，プログラミング能力とは，デバッグ能力ではないでしょうか。バグが出た。すべてチェックしたはずなのに動かない。そうした最悪の事態を打開していく能力，それがプログラミングに必要な腕力だと思います。

平木　敬太郎(20)X68000ACE-HD, PC-6001/8801 福井県

●現在のBASIC，とくにturboBASICは非常に強力で，マシン語っぽいコマンドもあったりして重宝しています。ときどきマシン語のプロトタイプを作成するために用いることも。「BASICの歴史と意義」を読みながら，インタプリタ型であるがゆえの欠点，利点などについて，いちいちうなずいてしまいました。マシン語に染まっているといっても，やはり私はBASICがけっこう好きなんですよね。特集で一番気に入ったのがハノイの塔のプログラムです。ちょっと変えて，結果をRAMディスクに落すようにしてN＝20でやってみたところ，45分走ったところで320Kいっぱいになり，DEVICE FULLで止まってしまいました。N＝64で世紀末が見えるというのもうなずけるな。　原　悟（18）X1turboII　宮城県

●僕にとってのBASICとはビデオのスローモーションのようなものである。BASICであるがために動きが（内部構造が）よくわかるのだ。そしてHuBASICのよいところは，命令が豊富，新しい機能にもすぐ対応できる，などだが，似たような命令（MUSIC/PLAY，SCREEN/GRAPHなど）を多くつけるなら，その分フリーエリアを広げてくれたらいいのに，とCZ-8FB01 Ver2.0を使っていると思う。

福島　義浩（19）X1turbo　滋賀県

ごめんなさいのコーナー

5月号　非BASICプログラマのためのMML

P.61　サンプルプログラムのテンポが少し速めでした。T38～40程度に変更してください。

5月号　あなたの知らない世界

P.79　X68000用のX1エミュレータではturbo用のBIOSが付属しませんのでX1turbo用アプリケーションは実行できません。

6月号　人類タコ科図鑑

P.29　ゼンジー北京は「中国は広島県生まれ」の誤植でした。

6月号　X68000用MACINTO-C

P.59　画面更新中に数値キーを押すと誤動作がありました。

317　bne　Cont

に変更し，318行に新しく，

318　Back：

というラベルの行を追加し，リスト1のプログラムを末尾に加え，また，25行目のデータは$0d，$0a，0に変更してください。

6月号　愛読者特大プレゼント

P.65　通信ソフトXLINKの価格は19,800円の間違いでした。お詫びして訂正いたします。

6月号　狂気のこきりこ

P.82　このプログラムの実行には1987年9月号のMML拡張が必要です。

6月号　ALAN

P.129　上下キーを入れ換える際のアドレスに誤りがありました。

CB60H → CB63H

に修正してください。

6月号　ふらっぺ

P.156　リストに一部欠けたところがありました。リスト2を追加してください。

リスト1

```
*変更
NextCont:
        add.l   #128,Pointer
        bsr     CheckLen
        bra     BlockOut
PrevCont:
        sub.l   #128,Pointer
        bra     BlockOut
Cont:
        cmp.b   #"g",d0
        beq     NextCont
        cmp.b   #"G",d0
        beq     NextCont
        cmp.b   #"t",d0
        beq     PrevCont
        cmp.b   #"T",d0
        beq     PrevCont
        bra     Back
        .END
```

リスト2

```
5870 DATA "! +-+ +-+ +-+ ++ !"
5880 DATA "!U   H! !M T U  U!"
5890 DATA "+-----+ +---------+"
5900 ' - 5 ﾒﾝ -
5910 DATA 2,2,2
5920 DATA 3
5930 DATA 32,12,2
5940 DATA 8 ,10,2
5950 DATA 14,6 ,1
5960 DATA 12,500
```

バグに関するお問い合わせは
☎03(263)2230(直通)
月～金曜日16：00～18：00

お問い合わせは原則として，本誌のバグ情報のみに限らせていただきます。入力法，操作法などはマニュアルをよくお読みください。
また，よくアドベンチャーゲームの解答を求めるお電話をいただきますが，本誌ではいっさいお答えできません。ご了承ください。

プログラム大募集！ Oh!X からの誘惑

▼あちこちで梅雨の季節に入りましたが，皆さんの腕力には影響ないですよね？ Oh!Xでは投稿プログラムを歓迎しています。ピコピコゲーム，大作ゲーム，ツールやユーティリティ，ミュージックプログラム，ショートショートなど，どしどし応募してください。6月号に掲載されたALANなど，最近はとくにMZ-2500ユーザーの活躍が目立っています。このところおとなしいX1ユーザーの皆さん，負けずにがんばりましょう。怒濤の投稿係はどんな作品でも受け付けます。

なお，応募の要領は，このページの右端の欄を参照してください。

▼エンドユーザーの立場から，望ましいパソコン環境について考えてきた「よりよいソフトウェア環境のために」は今月で連載を終了します。1年間，応援ありがとうございました。Macintoshの大好きな多摩豊氏にまた会える日を期待しましょう。

▼マニアックなファンの多かった「人類タコ科図鑑」も，残念ながら最終回です。おっと，石を投げないでください！ 祝一平氏は今月から「C調言語講座PRO-68K」をスタートします。特集「実践C言語からの誘惑」についても含め，皆さんの意見，感想，提案などを編集室にお寄せください。

▼定期購読のお申し込みや，バックナンバーのご注文についてのお問い合わせをよくいただきます。本誌でも，「バックナンバー案内」，および「編集室から」の最終ページにて，手続きの方法などをご案内しておりますので，そちらも参照してください。また，バックナンバーが在庫切れとなりました場合，コピーサービス等はいたしかねますので，恐縮ですがあらかじめご了承くださるようお願いいたします。

▼狭いとはいえ南北に長いわが国で，気候の話をすると，あちこちでインコンパチブルになってしまいます。季節の変わり目だからって夏風邪なんぞをひくとたいへん。冬と違ってマスクするわけにいかないし。えっ？ だってマスクって便利でしょ，堂々と顔が隠せるんだもん。眼鏡もかけてると耳が痛くなるのが少々悲惨だけど。とにかく健康には気をつけましょう。ではまた来月。

投稿応募要領

- 原稿には，住所・氏名・年齢・職業・連絡先電話番号・機種・使用言語・必要な周辺機器・マイコン歴を明記してください。
- プログラムを投稿される方は，詳しい内容の説明，利用法，できればフローチャート，変数表，メモリマップ（マシン語の場合）に，参考文献を明記し，プログラムをセーブしたテープ（ディスケット）を添えてお送りください。また，プログラムは最低2回はセーブしてください。
- ハードの製作などを投稿される方は，詳しい内容の説明のほかに回路図，部品表，できれば実体配線図も添えてください。編集室で検討の上，製作したハードが必要な場合はご連絡いたします。
- 投稿者のモラルとして，他誌との二重投稿，他機種用プログラムを単に移植したものは固くお断りいたします。

あて先

〒102 東京都千代田区九段南2-3-26井関ビル
日本ソフトバンク出版部
Oh！X「㋢㋮㋟」係

S H I F T ・ B R E A K

▶アスリートというスポーツドリンクを知ってますか？ なんと紅茶ベースのドリンクなんです。おもいっきり喉が渇いているときに飲んでしまった私は，あまりの味に30分ほど死んだうえにまるまるひと月，紅茶恐怖症になってしまいました。私でさえこうなるのだから普通の人だと……想像がつかん。さあ，いっぺん飲んでみたまい。（で）

▶エーン，エーン。即戦力の文書ディスクがぶっとんだよー。それも，今月号のレビューと，今書いている編集後記を，プリンタに打ち出す前にぶっとんじゃったんだよぉー。ひっく，ひっく。ちゃんとプロテクシトールを貼ってたのにー。それと私の文書ディスクを使って巻き添えをくった（で）さん，ごめんなさい。（悲しみにひれ伏すH.K.）

▶春が来て，とうとう我がサークルにも新入生が入ってきました。みんな若い！ 元気！ 先輩って大変なんだなぁと痛感するこのごろです。ところで7月2日筑波サーキットで行われる「筑波サタデージムカーナ」に参戦します。ひまな人は応援しに来るように。しかし愛車アルシオーネは……。（またしても被害にあったC.W.）

▶暇になるとWERDNAをやっつけては帰ってくる。気が向けばAMULET of WERDNAを20個ぐらい持ち帰ってみたり，どんどんクラスチェンジして年寄りだらけにしてみたり，それに飽きると若返らせてやったり，おそ松君ごっこしたり，地下11階を歩き回ったりして遊ぶ。僕の醒めた顔を想像してみてくれ。（II以降がやりたかった2500ユーザーのT.M.改めMu）

▶ザンギリ頭を叩いたら音がして以来100年以上たったわけだが，未だに情けないのはスーツを着こなしているサラリーマンをほとんど見ないことだ。ネクタイだってきちんと締めていない（つまり曲がっていたりずれていたり）人が多い。ワイシャツの一番上のボタンが留められないなんてみっともないわ暑苦しいわで，だからサラリーマンは嫌いだ。（K）

▶私は昨日，とある会社の社長さんから，非定型サイズでいかにも情けない風情の茶色い封筒を買った。なんでも「編集室に出入りしている人はみんな買っているよ」という口車にのせられて買ってしまったのだが，本当だろうか。事務所に遊びに行った栾野さんは袋貼りまで手伝わされたというし……。ねえK.Oさん，本当ですか？（K.S.）

▶少し前のこと，編集室で祝一平氏がこっそりと何かをプリントアウトしていた。思えば，あれが本誌6月号のあの広告の原稿だったのだ（何も教えてくれないんだから）。今，僕の手には創刊号がある（定期購読の予約をしたんだよ）が，ディスクを包む厚紙の「質実剛健」には笑ってしまった。でも僕は「逆襲のシャア」は好きですよ。（KO）

▶今日銀行でキャッシュカードを使おうとしたら，前の女の人がなかなか終わらない。どうしたんだろうと思って見ると，どうやら何度も限度額一杯の現金を引き出しているらしいのです。おそらくは1千万円以上になった頃，その機械は"打ち止め"になり，その人は紙袋を抱えて隣の引き出し機の列に並んだのでした。東京って怖いとこですねー。（M）

▶「次の誕生日でいくつになる？」「次……次の誕生日は，もうないんだ」酔っ払って暴れた男が，自分を捕らえた警官の問いにそう答える。見えすいた演出だとは思うが，癌で余命数カ月を宣告される建築家を演じたブライアン・デネヒーの最後の科白は，とてもこたえた。悪徳保安官でも宇宙人の親玉でも，とにかくこの俳優はすごいと思う。（よ）

▶先月の答えですが，1，3問目は主題歌に出るし，2問目は毎週出てきたセリフだし，ということで省略。難問は4問目で答えは「ラルフ」（アニメでは犬の名前）。全問正解者はただ者ではありません。2，3問正解の人は立派なオタクさんですね。なお，私の知人なら私を「オタク」とは呼ばないでしょう。（もっと無茶苦茶いわれそうな気もするU）

▶最近，12～13歳くらいの若い世代の方々からのハガキが多くなり喜んでいます。それと，これまで「X68000が欲しいが嫁さんがコワイ」と亭主族の攻撃の的にされていた若い奥様方からの反撃のお便りも多くなり，そのハガキを読むたびに大笑いしています。こんなパソコン雑誌ってあるのかな？ これからもこのような関係を大切にしたいものです。（N）

▶今月号は特集からしてCだったので，X68000よりの話が多くなってしまった。確かに，もっとX68000の記事をという要望は高いのだが，読者の数はX1ユーザーのほうが多いので，いつもこうだと期待されてもちょっと困る。また，8ビットユーザーにはもっともっとがんばってもらいたい。まあ，早く16ビットに乗り換えたい気持ちもわかるけどね。（T）

microOdyssey

5月の深夜,「魚が出てきた日」という映画がテレビで上映されていた。この映画は,核物質を積んだ飛行機が地中海に浮かぶ島に墜落,その前に乗務員とともにパラシュートでそこに投下された核物質の入った3つの箱を巡って島民や観光客,そして極秘に送り込まれた軍のエージェントたちが繰り広げる騒動を比較的軽いノリのコメディタッチで描いたものなのだが,しかしそのラストシーンは凄く,夜の海岸で若者たちや観光客がはしゃいでいるところに,羊飼いの手によって海中に捨てられた核物質の影響で魚が次々と海面に浮かび上がり,「警告します」という冷静な男の声のアナウンスが流れるシーンで終わる。

この映画は1967年に製作されたもので,私はいちばん最初はテレビの映画番組(荻昌弘さんの月曜ロードショウ)で17〜18年前に観ている。そして,そのときこのラストシーンになんともいえぬ恐怖感を抱いたものだった。その後,このラストシーン見たさにリバイバル上映時には映画館にも足を運んだこともある。

あのころはいまよりもっと感受性が強く,柴田翔や庄司薫の世界にどっぷりと首まで浸かっていて,「アイスコーヒーにミルクを入れて,クルクルとストローでかき混ぜる瞬間が好き。こうして混ざっていくの見ていると人生を感じる」などという軟弱な会話文を読んでは感動し,喫茶店をはしごしてはアイスコーヒーをひたすらクルクルかき混ぜていたものである。

まっ,それはどうでもいいとして,原発事故が起きた当時は,ヨーロッパから輸入されるワインや乳製品の安全性が問われていたこともあった。しかし,それもいつしか忘れ去られ,最近では事故後のチェルノブイリの記録映画を撮った監督が死亡したというニュースが話題を集めたくらいのものだ。だが,最近になって再び輸入食品の放射能汚染問題が全国生協組合を中心とした市民レベルの任意団体,「全国生協組合員ノーモアチェルノブイリの会」の手によって取り上げられようとしている。

この会は,放射能汚染値に安全制限はないという考えを基に,汚染食品の影響が高いと思われる年齢が低い子供たちを守ろうと発足されたもので,現在,輸入食品の放射能汚染値を個々の商品に表示するなど,10項目の公開質問状を用意し厚生省に提出するための署名運動を行っている。それは,現状では輸入食品全体の被ばく線限度の暫定基準(現行では年間500ミリレムの1/3以内)を厚生省が設置しているものの,その基準をクリアして商品として家庭内に入り込んだときは,もう放射能汚染などということ自体が誰の目にもわからないものとなっているからだ。

今回のように,市民団体が提示した食品に関する問題というのは,明らかに我々の日常生活を脅かす存在であり,放射能の蓄積摂取量がどうのといったことが世間一般に表面化するころには,とても取り返しのつかない事態になっている場合が多い。先に述べた映画の話のように,魚が浮かび上がってからの警告がなんの意味を持たないのは恐ろしい現実である。しかし,その魚を食べていることすら気づかないといったことだけは決して見逃してはならない。 (N)

1988年8月号7月18日(月)発売

特集1 ひと夏の数値演算

特集2 MIDIプログラミング

全機種共通システム
マルチウィンドウエディタ WINER

新連載 Z80実践マシン語講座

バックナンバー常備店

東京	神保町	三省堂神田本店5F	03(233)3312
	〃	書泉ブックマートBI	03(294)0011
	〃	書泉グランデ5F	03(295)0011
	八重洲	八重洲ブックセンター3F	03(281)1811
	新宿	紀伊国屋書店本店	03(354)0131
	高田馬場	未来堂書店	03(200)9185
	渋谷	大盛堂書店	03(463)0511
	池袋	西武百貨店11Fブックセンター	03(981)0111
	〃	西武百貨店9F コンピュータ・フォーラム	03(981)0111
	町田	久美堂東急ハンズ店	0427(28)2783
神奈川	横浜	有隣堂横浜駅西口店	045(311)6265
	〃	有隣堂ルミネ店	045(453)0811
神奈川	藤沢	有隣堂藤沢店	0466(26)1411
	厚木	有隣堂厚木店	0462(23)4111
	平塚	文教堂四の宮店	0463(54)2880
千葉	柏	新星堂カルチェ5	0471(64)8551
	船橋	西武百貨店10Fブックセンター	0474(25)0111
	〃	芳林堂書店津田沼店	0474(78)3737
	千葉	多田屋千葉セントラルプラザ店	0472(24)1333
埼玉	川越	黒田書店	0492(25)3138
	川口	岩渕書店	0482(52)2190
茨城	水戸	川又書店駅前店	0292(31)0102
大阪	都島区	駸々堂京橋店	06(353)2413
京都	中京区	オーム社書店	075(221)0280
愛知	名古屋	パソコンΣ上前津店	052(251)8334
長野	飯田	平安堂飯田店	0265(24)4545
北海道	室蘭	室蘭工業大学生協	0143(44)6060

定期購読のお知らせ

Oh!Xの定期購読をご希望の方は,最寄りの郵便局にある払込用紙に,

口座番号 東京1-29307

加入者名 株式会社日本ソフトバンク

とご記入のうえ,**年間購読料6,500円**を添えてお申し込みください。その際,裏面の通信欄に「○年○月号よりOh!X定期購読希望」と忘れずに明記してください。なお,すでに定期購読をご利用いただいている方には,購読期限終了と同時にご通知申し上げますので,同封の払込用紙をご利用ください。

海外送付ご希望の方へ

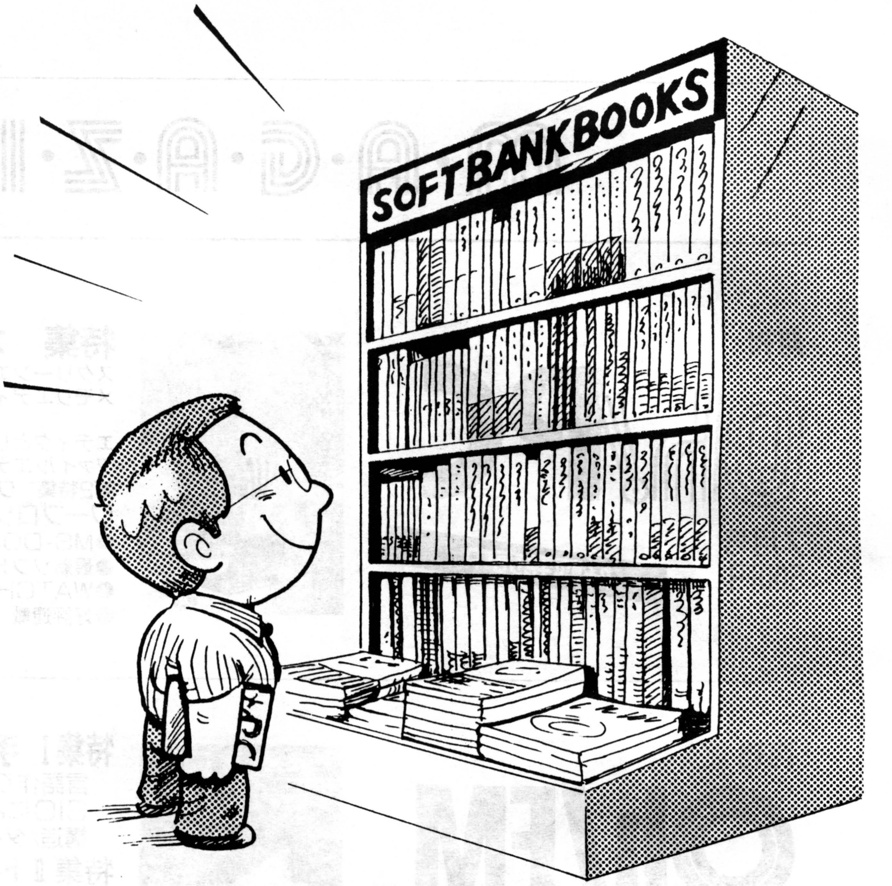
本誌の海外発送代理店,日本IPS(株)にお申し込みください。なお,購読料金は郵送方法,地域によって異なりますので,下記宛必ずお問い合わせください。

日本IPS株式会社
〒101 東京都千代田区飯田橋3-11-6
☎03(238)0700

Oh!X 7月号

■1988年7月1日発行 定価540円 **■発行人** 孫 正義 **■編集人** 笹口幸男
■発行元 (株)日本ソフトバンク
■出版事業部 〒102 東京都千代田区九段南2-3-26 井関ビル ☎03(261)4095 FAX 03(262)8397
編集室☎03(239)4156
出版営業☎03(261)4095
広告営業☎03(297)0181
■本 社 〒102 東京都千代田区九段南2-3-14 靖国九段南ビル ☎03(263)3690㈹
TELEX 東京 232-4614JSBTYJ FAX 03(263)3660
■西日本営業部 〒541 大阪府大阪市東区南本町2-6 明治生命堺筋本町ビル10F
☎06(264)1471 FAX 06(264)1481
■印 刷 凸版印刷株式会社

日本ソフトバンクの
書籍特約書店



下記の書店の一覧は、日本ソフトバンク書籍特約店として右にある商品の他、新刊もとりそろえております。ご希望の商品がある場合は、下記のお近くの書店にてお買い求め下さい。

(注)現品が売れて補充中の場合もございますので、ご注意下さい。

SOFT BANK

日本ソフトバンク出版事業部

〒102 東京都千代田区九段南2-3-26 ☎03(261)4095

全国特約書店一覧

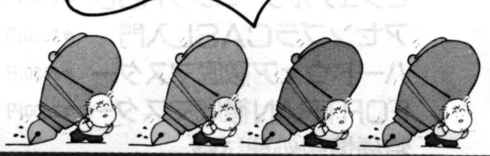
〈北海道〉		
札 幌 市	紀伊國屋書店札幌店	011-231-2131
〃	旭屋書店札幌店	011-241-3007
〃	丸善札幌支店	011-241-7252
〃	リーブルなにわ	011-221-3800
〃	冨貴堂札幌パルコ店	011-214-2303
〃	ダイヤ書房本店	011-712-2541
〃	ダイヤ書房西店	011-665-6223
旭 川 市	旭川冨貴堂	0166-26-3481
〃	ブックス平和マルカツ店	0166-23-6211
苫小牧市	旭屋書店苫小牧店	0144-36-5185
〈東　北〉		
青 森 市	成田本店	0177-23-2431
〃	岡田書店	0177-23-1381
弘 前 市	紀伊國屋書店弘前店	0172-36-4511
八 戸 市	伊吉書院	0178-44-1917
盛 岡 市	東山堂ブックセンター	0196-53-6464
〃	さわや書店	0196-53-4411
〃	第一書店	0196-53-3355
仙 台 市	金港堂	022-225-6521
〃	金港堂ブックセンター	022-223-0979
〃	アイエ書店駅前店	022-264-0718
〃	丸善仙台支店	022-266-1127
〃	高山書店	022-263-1511
〃	ブックスみやぎ	022-267-4422
秋 田 市	三浦書店	0188-33-8131
山 形 市	八文字屋	0236-22-2150
福 島 市	岩瀬書店コルニエツタヤ店	0245-21-2101
〃	博向堂	0245-21-1161
郡 山 市	東北書店	0249-32-0379
いわき市	ヤマニ書房本店	0246-23-3481
〃	鹿島ブックセンター	0246-28-2222
〈関　東〉		
水 戸 市	川又書店駅前店	0292-31-0102
〃	ツルヤブックセンター	0292-25-2711
勝 田 市	武石書店	0292-73-1212
鹿 島 郡	なみき書店	0299-96-1855
土 浦 市	共栄堂	0298-21-6134
筑 波 市	丸善筑波大学会館店	0298-51-6000
〃	友朋堂吾妻本店	0298-52-3665
宇都宮市	落合書店オリオン店	0286-34-3777
〃	落合書店東武ブックセンター	0286-34-8271
〃	新星堂宇都宮店	0286-33-2337
小 山 市	進駸堂駅ビル店	0285-25-1522
前 橋 市	煥乎堂	0272-23-1211
〃	リブロ前橋店	0272-34-1011
〃	戸田書店前橋店	0272-61-5063
高 崎 市	学陽書房	0273-23-4055
〃	サカヰ書店	0273-62-1500
〃	新星堂高崎店	0273-27-3961
〃	戸田書店高崎店	0273-63-5110
太 田 市	ナカムラヤ	0276-22-2001
〈首都圏〉		
浦 和 市	須原屋本店	0488-22-5321
〃	須原屋コルソ店	0488-24-5321
大 宮 市	押田謙文堂	0486-41-3141
〃	ブックランド押田	0486-47-3141
〃	三省堂ブックポート	0486-46-2600
蕨　　市	須原屋蕨店	0484-44-1211
川 口 市	岩渕書店川口店	0482-52-2190
川 越 市	黒田書店川越店	0492-25-3138
所 沢 市	芳林堂所沢店	0429-25-5355
〃	いけだ書店所沢店	0429-28-3271
上福岡市	黒田書店上福岡店	0492-66-0120
朝 霞 市	文教堂朝霞店	0484-76-0107
志 木 市	新星堂志木店	0484-74-0182
春日部市	文教堂春日部店	0487-52-7666
比 企 郡	錦電サービス	0492-96-2962
千 葉 市	多田屋セントラルプラザ店	0472-24-1333
〃	キディランド千葉店	0472-25-2011
習志野市	巌翠堂	0474-72-5011
船 橋 市	ときわ書房本店	0474-24-0750
〃	リブロ船橋店	0474-25-0111
〃	旭屋書店船橋店	0474-24-7331
〃	芳林堂津田沼店	0474-78-3737
〃	第二巌翠堂	0474-65-0926
柏　　市	西口アサノ	0471-44-2111
〃	新星堂柏店	0471-64-8551
松 戸 市	堀江良文堂	0473-65-5121
〃	辰正堂駅ビル店	0473-64-7997
横 浜 市	有隣堂トーヨー店	045-311-6265
〃	有隣堂東口ルミネ店	045-453-0811
〃	栄松堂相鉄ジョイナス店	045-321-6831
〃	そごうブックセンター	045-465-2111
〃	丸善ブックメイツポルタ店	045-453-6811
〃	有隣堂伊勢佐木店	045-261-1231
〃	有隣堂戸塚店	045-881-2661
〃	文華堂戸塚店	045-864-5151
〃	アーバン文華堂	045-821-5151
〃	文教堂青葉台南口店	045-983-5150
川 崎 市	有隣堂アゼリア店	044-245-1231
〃	有隣堂川崎BE店	045-261-1231
〃	文学堂本店	044-244-1251
〃	ブックセンター文教堂	044-811-5557
〃	文教堂溝の口店	044-811-8258
鎌 倉 市	島森書店大船店	0467-46-3841
〃	鎌倉書店	0467-46-2619
横須賀市	平坂書房WALK店	0468-25-5537
藤 沢 市	有隣堂藤沢店	0466-26-1411
〃	リブロ藤沢店	0466-27-0111
〃	文教堂六会店	0466-82-9610
茅ヶ崎市	川上書店ルミネ店	0467-87-3827
平 塚 市	サクラ書店駅ビル店	0463-23-2751
〃	文教堂四之宮店	0463-54-2880
小田原市	八小堂書店	0465-22-7111
〃	伊勢治書店	0465-22-1366
〃	文教堂小田原店	0465-36-3677
厚 木 市	有隣堂厚木店	0462-23-4111
大 和 市	文教堂中央林間店	0462-75-4165
相模原市	文教堂相模大野店	0427-49-0650
相模原市	文教堂橋本店	0427-74-5581
〃	文教堂星ヶ丘店	0427-58-6121
津久井郡	文教堂城山店	0427-82-9278
〈東　京〉		
千代田区	三省堂書店神田本店	03-233-3312
〃	書泉グランデ	03-295-0011
〃	東京堂書店	03-291-5181
〃	旭屋書店水道橋店	03-294-3781
〃	丸善お茶の水店	03-295-5581
〃	巌翠堂	03-291-1362
〃	いずみ神田南口店	03-254-8521
〃	明正堂秋葉原店	03-257-0758
中 央 区	八重洲ブックセンター	03-281-1811
〃	日本橋丸善	03-272-7211
〃	旭屋書店銀座店	03-573-4936
港　　区	書原新橋店	03-591-8738
〃	雄峰堂ＮＳ店	03-503-6586
〃	虎ノ門書房本店	03-502-3461
〃	虎ノ門書房田町店	03-454-2571
品 川 区	芳林堂大井町店	03-474-4946
〃	明屋書店五反田店	03-492-3881
渋 谷 区	紀伊國屋書店渋谷店	03-463-3241
〃	旭屋書店渋谷店	03-476-3971
〃	三省堂書店渋谷店	03-407-4545
〃	大盛堂書店	03-463-0511
〃	紀伊國屋書店笹塚店	03-485-0131
新 宿 区	紀伊國屋書店本店	03-354-0131
〃	三省堂書店新宿西口店	03-343-4871
〃	福家書店センタービル店	03-345-1246
〃	福家書店野村ビル店	03-342-0298
〃	新星堂ＮＳビル店	03-344-2055
〃	西武新宿ブックセンター	03-208-0380
〃	芳林堂高田馬場店	03-208-0241
〃	未来堂	03-200-9185
豊 島 区	旭屋書店池袋店	03-986-0311
〃	芳林堂池袋店	03-984-1101
〃	リブロ池袋店	03-981-0111
〃	三省堂書店池袋店	03-987-0511
〃	新栄堂本店	03-984-2345
〃	新栄堂アルパ店	03-988-0181
台 東 区	明正堂中通り店	03-831-0191
墨 田 区	リブロ錦糸町店	03-846-0111
〃	ブックストア・談	03-635-1841
江 東 区	新栄堂亀戸駅ビル店	03-638-2345
江戸川区	文教堂西葛西店	03-689-3621
大 田 区	アクトブックスサンカマタ店	03-735-1551
〃	竜文堂大森駅ビル店	03-775-3851
中 野 区	明屋書店東京本社	03-387-8451
杉 並 区	ブックセンター荻窪	03-393-5571
〃	書原杉並店	03-313-4778
武蔵野市	紀伊國屋書店吉祥寺東急店	0422-21-5543
〃	弘栄堂吉祥寺店	0422-22-1031
〃	パルコブックセンター吉祥寺	0422-21-8122
調 布 市	真光書店	0424-87-2222
府 中 市	啓文堂	0423-66-3151

展示図書一覧

書名	価格
MS-DOSいたれりつくせり本	●1800円
プレイMS-DOS	●1900円
UNIX System V プログラマ・ガイド	●12000円
UNIX System V ユーザ・ガイド	●9800円
C言語の活用理解	●2000円
C言語の基礎知識	●2500円
C言語の応用50例	●2300円
Cプリプロセッサ・パワー	●2200円
Play the C 上巻	●1500円
Play the C 下巻	●1500円
8086アセンブリ言語	●2800円
8086マクロプログラミング	●2600円
ビギニングMUMPS	●2600円
マシン語マジックブックII	●2500円
マシン語プログラミングテクニック	●2000円
BASICによるプログラミングスタイルブック	●1800円
ソーティング・ノート	●1900円
BASICプログラムジェネレータ集	●2800円
98/88スモールビジネスプログラム集	●2500円
88デスクアクセサリ集	●2000円
IDOS活用ハンドブック	●2700円
DISK CHARGE追補版	●1800円
フロッピーディスクフル活用ガイド	●2300円
PC工作入門	●1800円
試験に出るX1	●2800円
X1テクニカルマスター	●2500円
X1システム研究室	●2500円
新松ガイド	●2000円
一太郎Ver.3ガイド	●2500円
新一太郎ガイド	●2300円
一太郎ガイド	●2000円
桐Ver.2ガイド	●2500円
花子応用ガイド	●2500円
Lotus 1-2-3ガイド	●2400円
RDBファラオガイド	●2900円
ビジュアルラーニングRDB	●2500円
アセンブラCASL入門	●2000円
ハードウェア徹底マスター	●2500円
FORTRAN徹底マスター	●2800円
特種情報処理試験総整理と徹底対策	●2300円
情報処理の基礎知識	●1600円
ワープロ文書F・O・P	●1200円
新聞記事ハイテク切抜き法	●1200円
バイト&ワードの風にのって	●1800円
ワープロ考現学	●1200円
電子ゲームの「快楽」	●1200円
ムーグ・ノイマン・バッハ	●1300円
RPG幻想事典	●1500円
新明解ナム語事典	●5000円
保存版GS倶楽部	●1900円

三鷹市	三省堂書店三鷹店	0422-48-4510
〃	東西書房	0422-46-0275
小金井市	文教堂小金井店	0423-86-0161
国分寺市	三成堂国分寺店	0423-25-3211
国立市	東西書店	0425-75-5061
小平市	文教堂小平店	0423-43-9229
東村山市	文教堂東村山店	0423-96-1115
立川市	オリオン書房ウイル店	0425-27-2311
八王子市	くまざわ書店本店	0426-25-1201
町田市	有隣堂町田店	0427-23-3018
〃	久美堂本店	0427-25-1330
〃	久美堂小田急店	0427-27-1111
〃	久美堂東急ハンズ店	0427-28-2772
〃	文教堂鶴川店	0427-35-4117
〃	文教堂小川店	0427-96-1781
多摩市	くまざわ書店桜ヶ丘店	0423-37-2531
福生市	文教堂福生店	0425-53-7708
〈信越・北陸〉		
長野市	平安堂長野店	0262-26-4545
〃	長谷川書店	0262-26-2122
松本市	ブックスロクサン	0263-35-5555
〃	改造社松本駅ビル店	0263-36-3777
飯田市	平安堂飯田店	0265-24-4545
岡谷市	笠原書店	0266-23-5070
諏訪郡	平安堂諏訪店	0266-28-1111
新潟市	紀伊國屋書店新潟店	025-241-5281
〃	萬松堂	025-229-2221
〃	北光社	025-228-2321
長岡市	覚張書店	0258-32-1139
〃	ブックセンター長岡	0258-36-1360
〃	長岡技大長峰文化	0258-46-6437
富山市	瀬川書店	0764-24-4566
〃	清明堂	0764-24-4166
〃	BOOKSなかだ豊田店	0764-32-1353
〃	文苑堂本郷店	0764-22-0552
〃	文苑堂赤江店	0764-33-0321
高岡市	文苑堂	0766-21-0333
〃	文苑堂横田店	0766-21-0431
金沢市	うつのみや片町店	0762-21-6136
〃	書林香林坊本店	0762-20-5011
石川郡	王様の本野々市店	0762-46-5325
福井市	勝木書店	0776-24-0428
〃	品川書店新田塚店	0776-24-1112
〈東海〉		
静岡市	静岡谷島屋	0542-54-1301
〃	江崎書店	0542-54-4481
〃	吉見書店	0542-52-0157
〃	戸田書店SBS店	0542-81-5733
〃	戸田書店曲金店	0542-81-5899
沼津市	吉野屋	0559-23-5676
〃	マルサン書店宝塚店	0559-63-0350
富士市	戸田書店富士店	0545-51-5121
清水市	戸田書店本店	0543-65-2345
浜松市	浜松谷島屋	0534-53-9121
名古屋市	星野書店近鉄ビル店	052-581-4796
名古屋市	三省堂書店名古屋店	052-562-0077
〃	丸善ブックメイツセントラルパーク	052-971-1231
〃	日進堂上前津店	052-263-0550
〃	三洋堂パソコンショップΣ	052-251-8334
〃	三洋堂杁中店	052-832-8202
〃	ちくさ正文館本店	052-741-1137
〃	白樺書房西店	052-774-7223
豊橋市	精文館	0532-54-2345
岡崎市	ブックス鎌倉	0564-54-1822
刈谷市	三洋堂刈谷店	0566-24-1134
春日井市	三洋堂勝川店	0568-32-7806
岐阜市	自由書房	0582-65-4301
大垣市	大洞堂ブックス258	0584-81-2553
〃	大洞堂岐大バイパス店	0584-74-7766
可児市	三洋堂可児店	0574-63-2334
津市	別所書店11ビル店	0592-24-1014
四日市市	文化センター白揚	0593-51-0711
鈴鹿市	シェトワ白揚鈴鹿店	0593-82-5221
〈近畿〉		
京都市	駸々堂京宝店	075-223-1003
〃	アバンティ・ブックセンター	075-682-5031
〃	オーム社書店河原町店	075-221-0280
〃	ジュンク堂京都店	078-252-0050
奈良市	駸々堂大丸店	0742-26-6241
大阪市	旭屋書店本店	06-313-1191
〃	紀伊國屋書店梅田店	06-372-5821
〃	オーム社書店大阪店	06-345-0641
〃	駸々堂京橋店	06-353-3209
〃	駸々堂心斎橋店	06-251-0881
〃	旭屋書店ナンバ店	06-644-2551
〃	ナンバブックセンター	06-644-5501
〃	旭屋書店アベノ店	06-631-6051
〃	ユーゴー書店	06-623-2341
〃	河村書店	06-951-2968
枚方市	水嶋書房京阪デパート店	0720-51-3432
高槻市	コーベブックス西武高槻店	0726-83-1766
東大阪市	ヒバリヤ書店本社	06-722-1121
神戸市	ジュンク堂センター街店	078-392-1001
〃	ジュンク堂サンパル店	078-252-0777
〃	海文堂書店	078-331-6501
〃	日東館書林	078-391-8701
姫路市	新興書房	0792-85-3344
〃	誠信堂書店	0792-81-2055
和歌山市	宮井平安堂	0734-31-1331
〃	帯伊書店	0734-22-0441
〈中国〉		
岡山市	紀伊國屋書店岡山店	0862-32-3411
〃	丸善岡山支店	0862-31-2261
津山市	津山ブックセンター	08682-6-4047
広島市	紀伊國屋書店広島店	082-225-3232
〃	丸善広島支店	082-247-2251
〃	金正堂	082-248-3715
〃	積善館	082-248-3151
福山市	啓文社福山店	0849-22-3111
〃	ブックシティ啓文社	0849-25-0050
福山市	啓文社コア	0849-41-0909
山口市	五十部誠文堂	0839-24-6630
〃	文栄堂	0839-22-5611
宇部市	京屋書店	0836-31-2323
防府市	誠文堂国街店	0835-25-1988
鳥取市	富士書店	0857-23-7271
松江市	園山書店	0852-21-4167
〈四国〉		
徳島市	小山助学館本店	0886-54-2135
〃	小山助学館東口店	0886-25-1380
〃	森住丸善	0886-23-3228
高松市	宮脇書店本店	0878-51-3733
丸亀市	宮脇書店丸亀店	0877-22-5533
松山市	紀伊國屋書店松山店	0899-32-0005
〃	明屋書店本店	0899-41-4141
〃	明屋書店大街道店	0899-41-4242
〃	丸三書店	0899-31-8501
宇和島市	明屋宇和島店	0895-23-1118
高知市	金高堂	0888-22-0161
〈九州・沖縄〉		
福岡市	紀伊國屋書店福岡店	092-721-7755
〃	りーぶる天神	092-713-1001
〃	福岡金文堂	092-741-2106
〃	積文館新天町店	092-781-2991
〃	金文堂朝日ビル店	092-431-1094
北九州市	ナガリ書店	093-521-1044
〃	金栄堂	093-531-3685
〃	旭屋書店北九州店	093-631-6421
〃	井筒屋ブックセンター	093-641-0131
〃	カルパーク平野	093-661-7988
〃	白石書店本城店	093-601-2200
久留米市	エマックスたがみ	0942-33-1841
飯塚市	ブックリード	0948-25-7266
大分市	パルコブックセンター大分	0975-35-0643
〃	本町晃星堂	0975-33-0231
別府市	明倫堂	0977-23-0936
宮崎市	田中書店中央店	0985-24-5111
〃	宮崎寿屋	0985-27-4111
佐賀市	金華堂北バイパス店	0952-32-1965
〃	積文館デイトス店	0952-23-7155
長崎市	メトロ書店	0958-21-5453
〃	好文堂	0958-23-7171
佐世保市	金明堂	0956-22-4214
熊本市	紀伊國屋書店熊本店	0963-22-5531
〃	BOOKSまるぶん	0963-52-5665
〃	長崎書店	0963-53-0555
人吉市	明屋人吉店	0966-22-5486
鹿児島市	春苑堂ブックプラザ	0992-22-2131
〃	ブックスみすみ	0992-57-1011
那覇市	球陽堂ビル店	0988-63-3752
〃	文教図書	0988-62-1201

その筋に御用心



X1の美味しい機能をもりもりと料理

絶賛発売中　増刷出来！

試験に出るX1
ハードウェアのフルコース

祝　一平　著

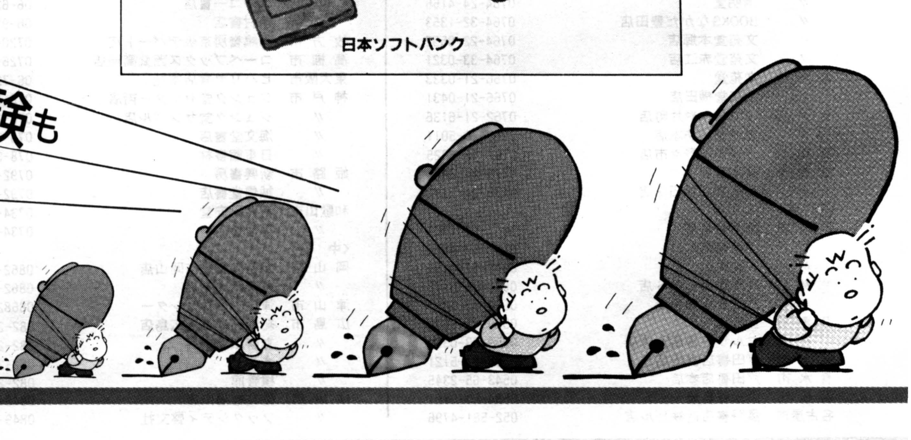
B5判　定価2,800円

Oh！MZ（1985年6月号～1987年8月号）に連載されたあの祝一平氏の「試験に出るX1」がついに1冊の本として完成しました。本書ではX1/X1turboシリーズのハードウェアをくまなく探検，筆者独自の解析術と豊富なオリジナルプログラムで数々の機能を料理していきます。連載時の内容にX1turboZの機能（第14章）を加筆，その他の章についても全面的に新情報を取り入れて再編集いたしました。さらに巻末には付録として「X1処理技術者試験」も収録しています。また，現在Oh！X掲載のミュージックプログラムで活用されているFM音源用MMLはX1ユーザーの必須アイテムと言えるでしょう。



内容

X1処理技術者試験もやってしまうのである。



SOFT BANK　発行
株式会社日本ソフトバンク出版事業部
〒102 東京都千代田区九段南2-3-26 ☎03(261)4095

本物かどうかが超多機能の条件。



SG SOFTWARE LIBRARY
SGソフトウェアライブラリー

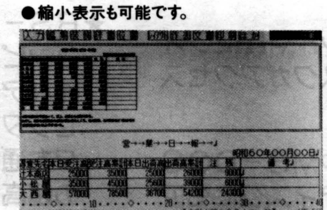
16ビット用最新、自動/一括/連文節変換システムKatana(刀)の完全移植。143万種にも及ぶ多彩な文字表現。※1 本格的データベース、表計算機能搭載。16ビットワープロソフト、データベースソフトなどMS-DOS上で動くソフトとのデータ互換。※2 その他すべての機能が16ビット用に開発されたパーツ群により構成。フルスペックでなおかつ超高速。

※1. 文字サイズ・文字種・文字の位置・網かけ・下線・カラー設定の組みあわせによる計算。※2. MS-DOSとのデータ交換は2HD版のみ。※MS-DOSはマイクロソフト社の登録商標です。

Katana(刀)が自動・一括・連文節変換実現。

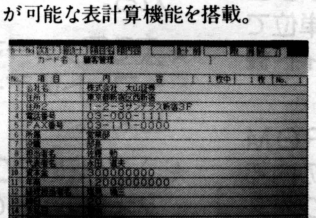
サムシンググッドが16ビット機上で開発した変換システムKatana(刀)を8ビット機用にコンバート。8ビットで初めて自動変換・一括変換・連文節変換を可能にしました。右の写真のような文章も一気に漢字かなまじり文に変換します。
しかもKatana(刀)の大きな特長は、品詞分類のきめ細かさと、独自の評価点数法を確立したこと。品詞をこれまでの倍以上(当社比)に分類し、かつ文節と文節のつながり方の妥当性を評価点によって判定することにより、既存の16ビットワープロソフトにも勝る高い変換効率を誇ります。

●縮小表示も可能です。



カード型データベース機能、表計算機能搭載。

住所録、名刺管理、カセットライブラリーなど使いみちタップリのデータベースと、行内・列内・行間・列間と多彩な計算が可能な表計算機能を搭載。



他の追従を許さぬ文字表現力。

文字のサイズは、1/4角から横4倍縦2倍角まで15種類。すべてのサイズの文字を、強調文字、白黒反転文字、斜体文字、袋文字に変換することが可能。これらの機能は、漢字・かな・記号など文字の種類を問いません。

多様な用紙への印刷が可能です。

はがき、原稿用紙、タックシールへの印刷を簡単に行うために専用の用紙設定を用意いたしました。

大好評発売中!!

超多機能日本語ワープロ

Shogun (将軍)

SHARP X1 turbo III / Z 専用2HD版
SHARP X1 turbo シリーズ対応2D版
※本商品はX1ではお使いいただけません。あらかじめご了承ください。

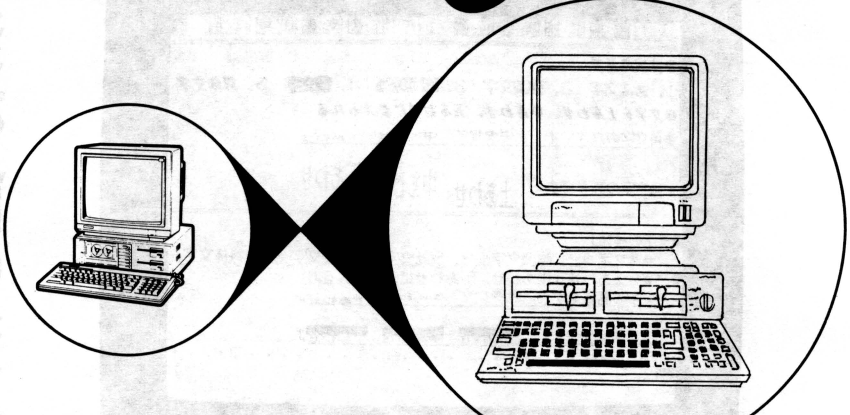
2D版、2HD版ともに ¥34,800

※Shogun(将軍)の画面デザイン・仕様等は改良を目的に予告なく変更する場合がございます。あらかじめご了承ください。
※Shogun(将軍)は、フロッピーの種類およびハードウェアのメモリ容量によって機能に違いがあります。あらかじめご了承ください。
〈既戦力〉X1turboシリーズ用をお使いの方はShogun(将軍)へのシステムアップサービスがございます。くわしくは弊社営業部までお問いあわせください。

人を大切にするテクノロジー
株式会社 サムシンググッド
〒160東京都新宿区大久保2-5-20シティプラザ新宿3F TEL03(232)0801(代表)
※資料のご請求は右の券を切りとり上記の弊社営業部宛までお送りください。
カタログ等お送りいたします。

資料請求券 oh!X Ⓢ 7月号

超高速の橋出現！



X1・X1turbo用

SUPER DEVICE MONITOR “T”

BLUE SKYはコンピュータ通信にオブジェクトデータの橋を架けました。今迄はRS-232Cでオブジェクトデータを通信する時は，アスキーデータに変換して行っていたコンピュータ通信を，直接オブジェクトデータのままで，しかも，特殊なデータ圧縮を施して，今迄にない超高速で通信する事が出来るX1turbo用の『SUPER DEVICE MONITOR “T”』を開発しました。既に好評発売中のMZ用の『SUPER DEVICE MONITOR “T”』とはRS-232Cにより双方向の超高速通信が出来ます。

エディット機能も呼び出したセクターを豊富なコマンドを使ってワープロ感覚で自在に変更・書き込み等のデータの編集が簡単に出来ます。アクセス出来るディバイスもハード・ディスク，MS-DOSやX68000で使用しているフォーマットの2HDのディスクなど各コンピュータに接続された殆どのディバイスをエディットする事が出来ます。

★任意のディバイスから他のディバイスへセクター単位で高速転送が出来る。

★任意のセクターをほぼ瞬間的に縦・横チェックサムとキャラクタダンプ付き表示が出来る。

★エディット機能はワープロ感覚で表示したセクターのオブジェクト・データを1バイト単位で変更・複写等多彩なエディット機能を備えている。

★*turbo*内のBIOS用ROMや*turbo*ZII標準装備の内部増設メモリーにも直接アクセス出来る。 （*turbo*のみ）

★任意のディバイスの複数のセクターを他のディバイスと比較・照合が出来る。

★キャラクターダンプは漢字の表示も出来る。 （*X1*は除く）

★RS-232Cのボーレートの変換はボタン一つで切り替えられる。

★X1フォーマットやMZフォーマットのディスクがアクセス出来る。

★X68000やMS-DOSフォーマットのディスクにもアクセス出来る。（*turbo*のみ）

★255バイト迄のデータを任意のディバイスの複数のセクターから検索する事が出来る。

★キャラクターダンプで表示出来る漢字には区点・JISの表示も出来る。 （*turbo*のみ）

★2HD及び2DDのディスクもアクセス出来る。（*turbo*のみ）

★RS-232Cを使っして他のコンピュータとの間で相互に特殊なデータ圧縮法に因り複数のセクターのオブジェクト・データを通常の最高32倍（理論値）の超高速での転送が出来る。 （*X1*は除く）

SUPER DEVICE MONITOR “T”	X1	5″	2D	10,000円
（*turbo*用の2HDは受注生産）	X1turbo	5″	2D／2HD	13,000円
	MZ-2500・2800	3.5″	2DD	13,000円

ロードに長時間かかる多分割のテープ版のゲームがボタン操作一つで何本も1枚のディスクに整理が出来て表示したリストから遊びたいゲームを指定すると一瞬でロード出来る『EXTRA HYPER＋α』もあります。

EXTRA HYPER ＋ α	X1・X1turbo	3″・5″	
	MZ-2000・2200・2500	3.5″・5″	各14,000円

BLUE SKY Co.

▶**お求めは全国の有名マイコンショップでどうぞ。**

通信販売をご希望の方は当社へ直接、商品名・機種名・メディア名・住所氏名・電話番号を明記の上、現金書留にてお申し込みください。（送料無料）

株式会社 BLUE SKY

〒411 静岡県三島市加茂16-4

☎0559-72-6710

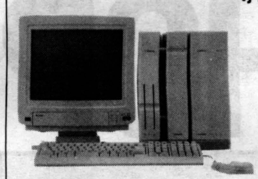
信用と実績を誇る BASIC HOUSE

北関東最大の68000専門店

SHARP SONY NEC 横河・ヒューレット・パッカード YHP Apple Computer

BASIC HOUSEで68000CPUが大流行

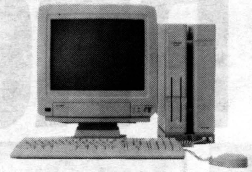
SHARP X68000（パーソナルワークステーション）
X68スーパーコブラII
BASIC HOUSE オリジナルセット
2MBバージョン
本体（増設メモリ内蔵）
専用ディスプレー



標準価格 ¥530,800
超特価 ¥398,000

長期クレジットOK 送料2,000

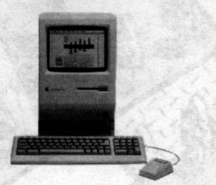
SHARP X68000 ACEHD
20MBハードディスク内蔵版
NEW 20MHD内蔵



本体CPU
専用ディスプレー
チルトスタンド
標準価格 ¥525,400
特価¥478,000

長期クレジットOK 送料2,000

Apple Macintosh Plus.
NEW 2MB
漢字Talk Ver.2.0
2MBメモリ内蔵



標準価格 ¥428,000
超特価
20Mハードディスク付
¥428,000

長期クレジットOK 送料2,000

BASIC HOUSE X68000 オリジナルハードウェア・ソフトウェア新製品

好評発売中	近日発売予定	好評発売中	好評発売中	好評発売中
型番	型番	型番	型番	型番
KGB-X681MB	KGB-X68PRK	KGB-X68ADC X	KGB-X68PIO	KGB-X68DAC
1MB増設メモリ ●ACEHD版等は使用できません。	数値演算プロセッサ 4MB増設RAMボード ●数値演算プロセッサはソケットのみ増設メモリーは1MB実装。	12Bit 16チャンネル 高速A/Dコンバータ サンプルソフト付	16Bit input 16Bit output 高級絶縁型PIO サンプルソフト付	12Bit 4チャンネル 高級D/Aコンバータ サンプルソフト付
定価¥32,000	定価¥58,000	定価¥128,000	定価¥68,000	定価¥118,000

新発売!!

●B6-6305……………… ¥6,800
C言語ライブラリー（XBASIC用）
B6-6301をお持ちの方はどうぞ!
●B6-6306……………… ¥14,800
BASIC拡張関数パッケージ C言語ライブラリー付
●KGB-X68UNB……… ¥6,800
X68000用ユニバーサル基板
金メッキ・スルホール・カードプラー付

BASIC HOUSEオリジナル

X68000シリーズ
●B6-6301……………………BASIC拡張関数パッケージ ¥9,800
●B6-6302……………………CP/M68K エミュレーター ¥19,800
●B6-6303……………………アイコンエディター ¥4,800
●B6-6304……………………ディスクキャシャー ¥6,800
●KGB-X681MB ……………1MB増設RAMボード（内蔵用） ¥32,000

MZシリーズ
●KGB-MZ 1 ………………超低価格計測制御ボード ¥15,500
●KGB-128KMZ…………MZ-2500用増設メモリボード ¥9,800
●ファミコンクリエーター………MZ-2500専用ファミコンソフトの解析ツール ¥25,000

X1/X1turboシリーズ
●KGB-X1S………… 低価格アナログデジタル入出力ボード ¥19,800
●KGB-HD 1/F………X1turbo 専用ハードディスクインターフェースボード ¥16,000
●KGB-P10……………高級絶縁型パラレル入出力ボード ¥42,000
●KGB-AD12……………高級16ch 12Bit A/D変換ボード ¥118,000
●KGB-DA4……………高級4ch 12Bit D/A変換ボード ¥98,000
●B6-3301………………PC98↔X1turbo相互ファイルコンバーター ¥4,800

新発売 X1・1turbo用 GP-IBインターフェースボード
型番 KGB-488
（マニュアルソフト付）
定価 ¥58,000

X・MAC World Expo in NIKKO 開催

SHARP X68000 と Apple Macintosh
ユーザーズグループの合同コンパ
全国の好者が集まり、夜通し遊んでしまう。

開催予定日 昭和63年7月30日(土)・31日(日)
会場 奥日光高原キャンプ場
会費 ¥10,000
内容
● X68000関係セミナー（テーマ募集）
● Macintosh関係セミナー（Hypowercard）
● パブリックドメインソフト大集合
● 自作ソフトの交換会

定員40名ですので、御早めに!!

全国どこでも発送可 長期クレジットOK 送料全国均一¥1,000 宅配便にて即日配送

株式会社計測技研 本社営業部／マイコンショップ／通販部 宇都宮市竹林町503―1 TEL0286-22-9811 FAX0286-25-3970

マイコンショップ BASIC HOUSE お申し込み・お問い合せは ☎0286-22-9811(代)



当店はX68000の認定店です。どんなことでも安心して ご相談ください。
★X68000をお買上げのお客様にもれなくテレホンカードとゲームソフト(アルカノイド)をプレゼント中！

営業時間
AM10:00～PM7:00
(日曜・祭日はPM6:00まで)
年中無休

これからは16ビット！ X68000

START → お好きな組合せでどうぞ。

本体

スタンダードモデル
X68000
・CZ-601C(E・B) ¥319,800

プロフェッショナルタイプ
X68000 ACE HD
・CZ-611C-GY … ¥399,800
新製品・20Mハードディスク内蔵!!

ディスプレイ

● CZ-601D-GY ……… ¥119,800
ピッチ0.39・アナログ対応

● CZ-611D-GY ……… ¥145,000
ピッチ0.31・アナログ対応

● CU-15M1(E・B) …… ¥ 99,800
ピッチ0.39・アナログ/デジタルモニター

周辺機器・ボード

● CZ-8PK7 ……… ¥122,000
80桁ドットインパクトプリンター

● CZ-8PK8 ……… ¥152,000
136桁ドットインパクトプリンター

● CZ-8PC2 ……… ¥ 69,800
80桁熱転写プリンター

● CZ-6BE1 ……… ¥ 35,000
1MB増設RAM(CZ-600C用)

★その他いろいろあります。お電話で！

ソフト PART 1

● 日本語ワープロEW …… ¥38,000
フロントプロセッサE1搭載ワープロソフト

● WINDEX PRO-68K・¥28,000
コンパイラと来たらエディタです。

● Kamikaze ……… ¥68,000
忘れちゃいけないビジネスソフト

● Z'S STAFF PRO-68K・¥58,000
プロフェッショナルグラフィックツール

ソフト PART 2

● XLink 68 ……… ¥19,800
時代はパソコン通信だ！

● ミュージックPRO-68K … ¥18,800
● サウンドPRO-68K …… ¥15,800
ミュージック関係ならこの2本！

● サンプリングPRO-68K … ¥17,800
PCMをフル活用するならこれ！

● C-TRACE68000 …… ¥68,000
本格的レイトレーシングツール

GOAL

さあ、ご注文、お問合せは今スグお電話で！お支払いは超低金利のクレジットもご利用できます。お気軽にご連絡ください。

☎03-486-6541

ソフトやハードの内容や発売日等のおたずねにも親切にお答えします。

組合せのほんの一例

名づけて… 必殺！ゲーマーズセット

● CZ-601C(E・B)本体＋キーボード	¥319,800
● CZ-601D(E・B)ディスプレイテレビ	¥119,800
● CZ-6ST1(E・B)チルトスタンド	¥ 5,800
● スペースハリアー	¥ 6,800
● 源平討魔伝	¥ 7,800
● XE-1PRO(ジョイスティック)	¥ 9,500
■定価合計	¥528,700

均等払い	ボーナス
¥17,140×18回	¥30,000× 3回
¥12,900×24回	¥25,000× 4回
¥ 8,650×36回	¥20,000× 6回

●ソフトも周辺機器も紹介しきれないぐらい豊富です。くわしくはお電話で！

それでもやっぱりX1だ！

X1 turbo Z II

● CZ-881C-BK本体＋キーボード	¥179,800
● CZ-880D-BKディスプレイテレビ	¥109,800
● CZ-6ST1B チルトスタンド	¥ 5,800
● AN-160SPアンプ内蔵スピーカー	¥ 59,800
● ブランクディスケット	¥ 4,500
■定価合計	¥359,700

X1 パソコンテレビ twin

● CZ-830C-BK本体＋キーボード	¥99,800
● CZ-830D-BKディスプレイテレビ	¥98,000
● CZ-6ST1B チルトスタンド	¥ 5,800
● 上海(ゲームソフト)	¥ 4,500
● ブランクディスケット	¥ 4,500
■定価合計	¥212,600

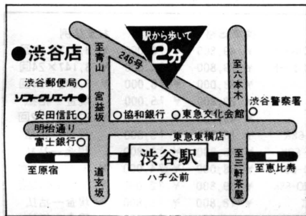
クリエイト特典

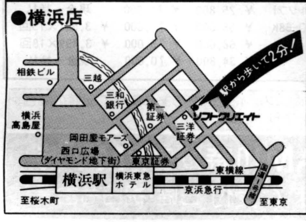
● 全商品完全保証書付(メーカー保証)
● 全国無料配達(一部離島の方は有料になります)
● 配達日の指定OK(日曜・祭日にかかわらずお客様のご都合にあわせて配達します)
● どんな商品の組合せも自由自在(ご予算、用途に応じ自由自在にシステムアップできます)
● 中古パソコン高額下取(今お使いのパソコンをわずかな差額でグレードアップ)
● お支払い方法自由(低金利の均等払、ボーナス一括払もご利用下さい)

大特価周辺機器(各ケーブル付き)

品名	定価	機能説明
ITH-320S	~~¥125,000~~	20Mハードディスク 平均シークタイム 28ms以下
ITH-520N	~~¥ 99,800~~	20Mハードディスク 平均シークタイム 65ms以下
ITH-540S	~~¥168,000~~	40Mハードディスク 平均シークタイム 38ms以下
VP-800	~~¥122,000~~	80桁シリアルプリンタ

夏のパソコン祭り 7/8(金)～7/10(日)
1. 各社パソコン、ワープロお祭り特価セール！
2. ゲーム＆ビジネスソフト大幅値引き！
3. 特別高額下取りセール！
4. フロッピィ、マウス等サプライ用品大特売！
5. 各社プリンタ、HD等周辺機器大特価セール！

●渋谷店


●横浜店


総合お問合せ先☎03-486-6541(代)

パソコン専門ショップ
ソフトクリエイト 渋谷/横浜

●渋谷店☎03-486-6541(代)
〒150:東京都渋谷区渋谷1-12-7 三和渋谷ビル
振込銀行:協和銀行 渋谷支店(普)No.239313

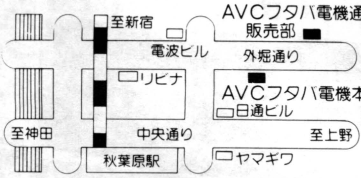
●横浜店☎045-314-4777(代)
〒221:横浜市神奈川区鶴屋町2-12-8 第1建設ビル
振込銀行:三和銀行 横浜駅前支店(普)No.310852

秋葉原価格でローンができます
電気の街秋葉原で 24年の信用!!

AVCフタバ

AUDIO VIDEO COMPUTER

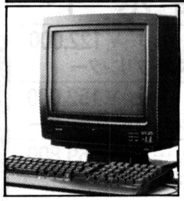
☎03(253)7661



AVCフタバ電機
〒101 東京都千代田区外神田2-9-8
神田ユニオンビル ☎03-253-7661(代)

今すぐ もよりの電話から	仙台 022-264-3704	名古屋 052-452-3271	広島 082-295-6873
札幌 011-611-5104	新潟 0252-75-4175	大阪 06-311-3931	福岡 092-481-2494

X68000 ACE-HD



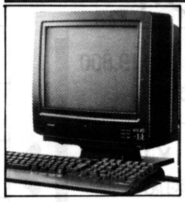
ドットピッチ0.31mmのモニターをセット20M HD搭載の超高級セット。

CZ-611C……¥399,800
CZ-611D……¥145,000
合計………¥544,800

特価 ¥4?4,000

お支払例 初¥16,660+¥14,900×35回
初¥14,300+¥11,700×35回

X68000 ACE-HD



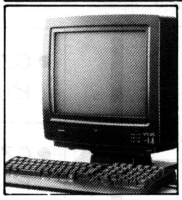
強力な日本語処理と実装密度を追求して信頼性、更に向上。

CZ-611C……¥399,800
CZ-601……¥119,800
合計………¥519,600

特価 ¥4?4,000

お支払例 初¥16,360+¥14,200×35回
初¥11,800+¥11,200×47回

X68000 ACE-HD



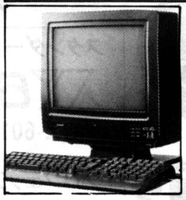
更に夢を拡大、20MB HDの搭載。最大に能力を引出す3モードのディスプレイ。

CZ-611C……¥399,800
CU-15MI……¥99,800
合計………¥499,600

特価 ¥4?0,000

お支払例 初¥16,500+¥13,700×35回
初¥12,400+¥10,800×47回

X68000 ACE



CZ-600の後継 直接アクセスできるメモリーが16MBもある優れもの。

CZ-601C……¥319,800
CZ-601D……¥119,800
合計………¥439,600

特価 ¥3?3,000

お支払例 初¥14,220+¥12,100×35回
初¥12,400+¥9,500×47回

X1turboZII



X1turboZの本格派セット。TV付2モードオートスキャンディスプレイ。

CZ-881C……¥179,800
CZ-881D……¥109,800
合計………¥289,600

特価 ¥2?3,000

お支払例 初¥10,520+¥7,600×35回
初¥7,900+¥6,000×47回

X1turboZII



NEW-ZBASICの搭載でAV機能をサポート、充分に楽しめるぞ。

CZ-881C……¥179,800
CU-14BD……¥64,800
合計………¥244,600

特価 ¥1?2,000

お支払例 初¥7,080+¥6,600×35回
初¥5,200+¥5,200×47回

X1turboZ



NEW-ZBASICは後で買えばいい。ハイグレードモニタをセットして驚異の価格。

CZ-880C……¥218,000
CZ-880D……¥109,800
合計………¥327,800

特価 ¥1?8,000

お支払例 初¥9,120+¥6,400×35回
初¥9,400+¥5,000×47回

X1turboZ



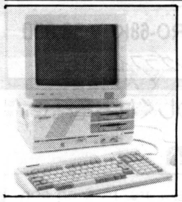
名機X1turboZが更に安くなりました。2モードディスプレーとの組合せ。

CZ-880C……¥218,000
CU-14BD……¥64,800
合計………¥282,800

特価 ¥1?3,000

お支払例 初¥8,040+¥7,500×23回
初¥7,720+¥5,200×35回

mZ-2861



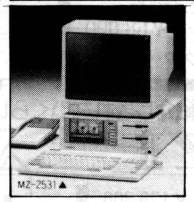
高級日本語ワープロ「書院28」搭載、MS-DOSと融合しスピーディな実務。

MZ-2861……¥328,000
MZ-1D26……¥89,800
合計………¥417,800

特価 ¥3?2,000

お支払例 初¥14,580+¥11,700×35回
初¥12,200+¥9,200×47回

mZ-2531



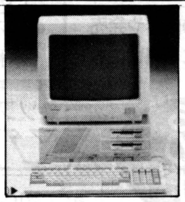
今よりもなおハイグレードに、とお考えの方に…更に可能性を拡大する。

MZ-2531……¥199,800
MZ-1D22……¥108,000
合計………¥307,800

特価 ¥2?8,000

お支払例 初¥11,620+¥8,100×35回
初¥8,600+¥6,400×47回

mZ-2520



MZをこれから使う方、少々ぜいたくですが是非お勧めしたいセットです。

MZ-2520……¥159,800
MZ-1D26……¥89,800
合計………¥249,600

特価 ¥2?4,000

お支払例 初¥7,960+¥7,000×35回
初¥6,700+¥5,500×47回

X1twin



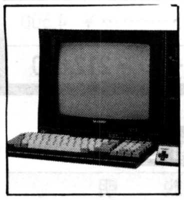
HEシステムを搭載、最上級ゲーム機とパソコンが合体。

CZ-830C……¥99,800
CZ-830D……¥98,000
合計………¥197,800

特価 ¥1?0,000

お支払例 初¥5,900+¥5,500×35回
初¥5,900+¥4,300×47回

X1Gmodel30



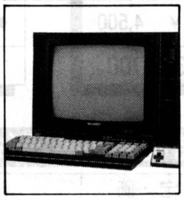
X1Gの本格派セットFDD2基内蔵、専用カラーモニタはTVにも使用可能。

CZ-822C……¥118,000
CZ-820D……¥79,000
合計………¥197,000

特価 ¥1?9,800

お支払例 初¥7,664+¥5,300×23回
初¥6,652+¥3,700×35回

X1Gmodel30



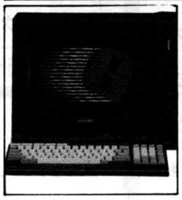
こちらはモニタまで予算がまに合わない方、RFコンバータ付きで家のTVで……。

CZ-822C……¥118,000
AN-58C……¥2,980
合計………¥120,980

特価 ¥?9,800

お支払例 初¥7,078+¥6,400×11回
初¥4,164+¥3,400×23回

X1Gmodel10



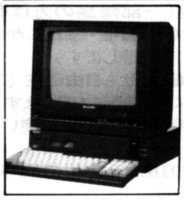
FDDも欲しい、レコーダも使いたい、予算が10万以下ならこのセットをお勧めします。

CZ-820C……¥69,800
CZ-820D……¥79,000
CZ-503F……¥49,800
合計………¥198,600

特価 ¥?4,800

お支払例 初¥6,064+¥4,600×23回
初¥5,552+¥3,200×35回

X1Fmodel10



とりあえずBASICを覚えてディスクソフトで遊ぶのは後まわし、そんな君へのセット。

CZ-811C……¥89,800
CU-14G……¥49,800
合計………¥139,600

特価 ¥59,800

お支払例 初¥7,280+¥6,500×9回
初¥4,768+¥3,400×19回

●分割回数は3回～48回まで自由に選べます。

●セットの組合せは自由!広告に出ていない他の機種はお問合せ下さい。

型番	品名	標準価格	販売価格	お支払例
CU-14GE	ディスプレイ	¥49,800	¥?9,800	¥3,278×10回
CU-14BD	ディスプレイ	¥64,800	¥?8,000	¥5,280×10回
CU-14A4	ディスプレイ	¥89,800	¥?3,000	¥3,371×18回
CU-14AD	ディスプレイ	¥84,800	¥?8,000	¥3,689×18回
CU-15MI	ディスプレイ	¥99,800	¥?7,000	¥3,786×24回
CZ-820D	ディスプレイ	¥79,800	¥?6,000	¥3,465×15回
CZ-880D	ディスプレイ	¥109,800	¥?3,000	¥4,081×24回
CZ-601D	ディスプレイ	¥119,800	¥?3,000	¥3,203×36回
CZ-611D	ディスプレイ	¥145,000	¥1?3,000	¥3,892×36回
CZ-520F	FDD(2HD/2DD)	¥118,000	¥?2,000	¥3,169×36回
CZ-502F	FDD(2DD)	¥99,800	¥?8,000	¥2,687×36回
CZ-503F	FDD(2D)	¥49,800	¥?8,000	¥3,515×12回
CZ-6BE1	1MB (増設RAMボード)	¥35,000	¥?8,000	¥3,080×10回
CZ-6BE2	2MB (増設RAMボード)	¥79,800	¥?2,000	¥3,944×18回
CZ-6BE4	4MB (増設RAMボード)	¥138,000	¥1?8,000	¥3,720×36回

型番	品名	標準価格	販売価格	お支払例
CZ-620H	HDD	¥178,000	¥1?2,000	¥3,846×48回
CZ-8PC2	プリンタ(80桁)	¥69,800	¥?4,000	¥4,995×12回
CZ-8PK5	プリンタ(80桁)	¥129,000	¥1?0,000	¥3,444×36回
CZ-8PK7	プリンタ(80桁)	¥122,000	¥?4,000	¥3,238×36回
CZ-8PK6	プリンタ(136桁)	¥159,000	¥1?3,000	¥3,331×48回
CZ-8PK8	プリンタ(136桁)	¥152,000	¥1?7,000	¥3,169×48回
CZ-8PD3	プリンタ	¥59,800	¥?6,000	¥3,465×15回
CZ-8NS1	カラーイメージスキャナ	¥188,000	¥1?0,000	¥4,063×48回
CZ-6VT1	カラーイメージユニット	¥69,800	¥?6,000	¥3,562×18回
CZ-6PV1	カラービデオプリンタ	¥198,000	¥1?4,000	¥4,171×48回
AN-160SP	アンプ内蔵スピーカー	¥59,800	¥?8,000	¥3,053×18回
CZ-8BS1	FM音源ボード	¥23,800	¥?9,800	現金一括払
CZ-6BN1	スキャナ用パラレルボード	¥29,800	¥?4,000	現金一括払
CZ-6BU1	ユニバーサルIOボード	¥39,800	¥?2,000	¥3,520×10回
CZ-6BG1	GP-IBボード	¥59,800	¥?8,000	¥3,053×18回

型番	品名	標準価格	販売価格	お支払例
CZ-6BF1	増設RS232Cボード	¥49,800	¥?0,000	¥3,013×15回
CZ-6BP1	数値プロセッサボード	¥79,800	¥?4,000	¥3,147×24回
CZ-6EB1	I/Oボックス	¥88,000	¥?8,000	¥3,343×24回
CZ-8TM1	モデム	¥29,800	¥?5,000	現金一括払
CZ-8TM2	モデム	¥49,800	¥?9,000	¥3,608×12回
CZ-213MS	MUSIC PRO-68K	¥18,800	¥?5,800	現金一括払
CZ-214MS	SOUND PRO-68K	¥15,800	¥?3,800	現金一括払
CZ-212BS	ビジネス PRO-68K	¥68,000	¥?4,000	¥3,435×18回
CZ-211LS	Cコンパイラ PRO-68K	¥39,800	¥?2,000	¥3,520×10回
CZ-141SF	NEW-Z BASIC	¥18,800	¥?5,800	現金一括払
CZ-137SF	turboZ's STAFF	¥19,800	¥?6,800	現金一括払
CZ-133SF	モデムターミナルソフト	¥25,800	¥?1,000	現金一括払
	Z' STAFF PRO-68K	¥58,000	¥?7,000	¥3,541×15回
	kamikaze	¥68,000	¥?5,000	¥3,499×18回
	Shogun	¥34,800	¥?0,000	現金一括払

頭金なし 手軽な電話クレジット。

カレッジクレジット 保証人なし。但し満20才以上の学生の方。

納期 通常の場合、当社に申込書が到着後1週間以内。特に人気のある商品で品薄の場合、少々納期が遅れる場合もありますので御了承下さい。

製品先取り お支払いは約1～2カ月後から。

低金利クレジット 1回の支払は2,700円以上で3～48回。ボーナス併用も可。

完全保証 すべてメーカー保証書付 アフターケア万全。

全国代引 お届けした者に、代金をお支払いいただく方法です。(但し、手数料1,000円)

18才未満の方 ご両親が代理購入者としてお申し込み下さい。

AM10時からPM7時まで受付 日曜・祝日も営業

さらに金利が
安くなった!!
超低金利クレジット

▶12回……4.5%
▶24回……9.5%
▶36回……13.0%
▶48回……17.0%
▶60回……22.0%

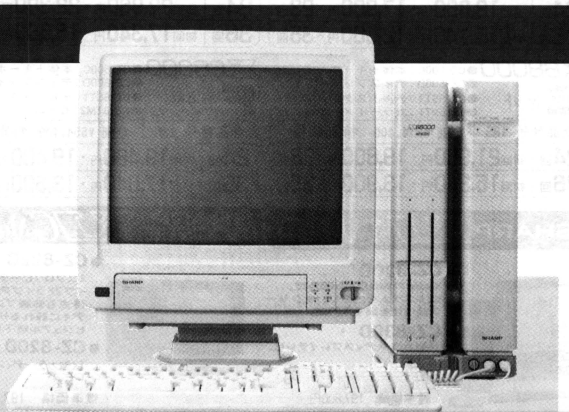
またまた秋葉原でおなじみの

6/15～7/20

冬のボーナス
一括払い手数料なし
(ボーナス2回払いもご利用下さい)

●お近くの方はお
●本体単品で特
●ビジネスソフト定

X68000ACE HD (送料￥2,000)



Ⓐセット:
CZ-611C＋CZ-611D＋M-2HD(10枚)
……定価￥544,800➡P&A超特価(価格はお電話下さい。)

12回	37,000	24回	19,300	36回	13,300	48回	10,300	60回	8,600

Ⓑセット:
CZ-611C＋CZ-601D＋M-2ND(10枚)
……定価￥519,600➡P&A超特価(価格はお電話下さい。)

12回	35,200	24回	18,400	36回	12,700	48回	9,800	60回	8,200

※X-68000セットでお買い上げの方に源平討魔伝￥7,800をプレゼント致します。
※チルトスタンド(CX6ST￥5,800)必要な方は￥5,000加算して下さい。

X68000ACE(送料￥2,000)



Ⓐセット:
CZ-601C＋CZ-611D＋M-2HD(10枚)
……定価￥464,800➡P&A超特価(価格はお電話下さい。)

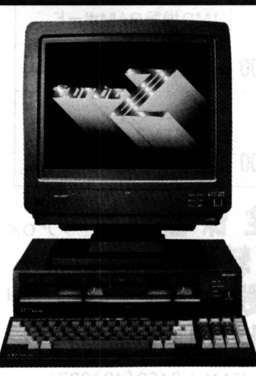
12回	31,300	24回	16,400	36回	11,300	48回	8,700	60回	7,300

Ⓑセット:
CZ-601C＋CZ-601D＋M-2HD(10枚)
……定価￥439,600➡P&A超特価(価格はお電話下さい。)

12回	29,600	24回	15,500	36回	10,600	48回	8,200	60回	6,900

※チルトスタンド(CZ-6STI ￥5,800)必要な方は￥5,000加算して下さい。
※X-68000セットでお買い上げの方に源平討魔伝￥7,800をプレゼント致します。

X-1ターボZ/ZⅡ (送料￥2,000)



※チルトスタンド(CZ-6STI￥5,800)必要な方は￥5,000加算して下さい。

Ⓐセット:
X.1ターボZ(CZ-880C＋CZ-880D)＋M-2HD(10枚)＋ジョイカード＋ゲームソフト3種
……定価￥327,800➡超特価**￥180,000**

12回	15,600	24回	8,200	36回	5,600	48回	4,300	60回	3,600

●NEW Z-BASIC(CZ-141SF￥18,800)必要な方は，￥15,000加算して下さい。

Ⓑセット:
X-1ターボZⅡ(CZ-881C＋CZ-880D)＋M-2HD(10枚)＋ジョイカード＋ゲームソフト3種
定価￥289,600➡P&A超特価(価格はTel下さい。)

12回	18,200	24回	9,500	36回	6,500	48回	5,100	60回	4,200

ソフトコーナー (送料￥1,000)

X- 68000用

Ⓐ CZ-213MS(MUSIC PRO-68K)……定価￥18,800➡特価**￥16,000**
Ⓑ CZ-214MS(SOUND PRO-68K)……定価￥15,800➡特価**￥13,500**
Ⓒ CZ-212BS(BUSINESS PRO-68K)……定価￥68,000➡特価**￥57,000**
Ⓓ CZ-211LS(C compiler PRO-68K)……定価￥39,800➡特価**￥33,500**
Ⓔ Z's STAFF PRO-68K(シャフト)……定価￥58,000➡特価**￥46,000**
Ⓕ Kamikaze(神風)(サムシンググッド)……定価￥68,000➡特価**￥49,000**
Ⓖ ビジネスAD68K(マッシュシステム)……定価￥98,000➡特価**￥78,500**
Ⓗ 弥生(日本マイコン販売)……定価￥80,000➡特価**￥64,000**
Ⓘ CP/M-68K(ニューウェイブ)……定価￥110,000➡特価**￥88,000**

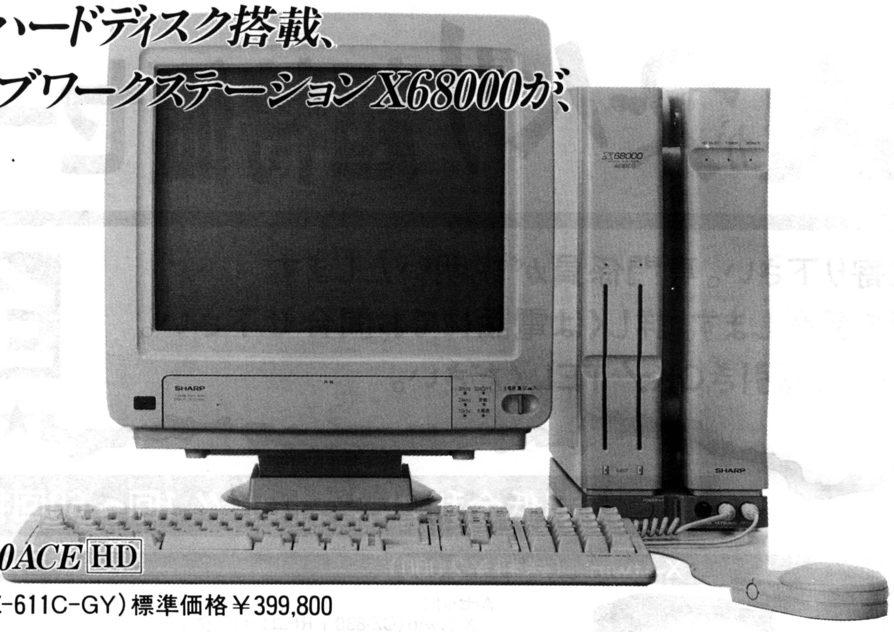
X-1シリーズ

Ⓙ SHOGUN(サムシンググッド)……定価￥34,800➡特価**￥25,000**
Ⓚ SAMURAI(サムシンググッド)……定価￥19,800➡特価**￥15,200**

価値あるアイビット、パワーアップ下取りセール！

いま御使用のパソコンを高価下取りの上、MZ-6500モデル50、シャープX68000ACE HD、またはパソコンテレビX1turboZをアイビットならではのサービス大特価でお届けします。

20Mバイトハードディスク搭載、クリエイティブワークステーションX68000が、いま熱い



シャープX68000ACE HD

本体＋キーボード（CZ-611C-GY）標準価格￥399,800

CZ-600Cを下取りした場合 特価￥178,000

●68000搭載●最大12バイトの大容量メモリ●20Mバイトハードディスク内蔵●高解度自然色グラフィックス●フレンドリーOS Human68K搭載等、先進機能満載。……豊富な周辺機器がサポート

セット大特価！ズバリ￥340,000！

●本体＋キーボード（CZ-600E） 標準価格~~￥369,000~~
●15型カラーディスプレイテレビ（CZ-600DE） 標準価格~~￥129,800~~
●チルトスタンド（CZ-6ST1E） 標準価格~~￥5,800~~

高度なパフォーマンスを秘めて、新登場！

12MHzの高速80286CPU、高速グラフィックLSI搭載。
AI辞書による高度な日本語処理、MS-DOS V3.1。



mz-6500 モデル50

●16ビットパーソナルコンピュータMZ-6551
（1.2MB FD2基搭載）標準価格￥430,000
●16ビットパーソナルコンピュータMZ-6556
（1.2MB FD2基、20MB HD1基搭載）標準価格￥650,000

☆下取り価格、及び特価につきましては、電話でお問い合わせください。

リアルな映像と音が創造力を刺激する。



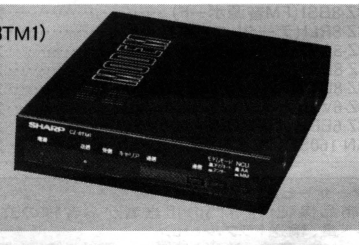
パソコンテレビX1turboZ
本体＋キーボード（CZ-880C）
標準価格￥218,000

下取り機種問わず/サービス大特価￥100,000

●アナログカラーイメージボード内蔵●4,096色対応ニューテロッパ機能●8重和音ステレオFM音源搭載●マウス標準装備●JIS第1/第2水準漢字ROM実装●システム、ユーザー辞書装備●1Mバイト5インチフロッピー2基搭載。

6月30日までに お買い上げの方に プレゼント

●モデムユニット（シャープCZ-8TM1）



AIBIT
アイビット電子株式会社
〒192 東京都八王子市北野町560-5

☎0426-45-3001～3
FAX.0426-44-6002

●営業時間：10:00～19:00
●電話受付：20:00迄可
●定 休 日：日曜日（祭日営業）

信用をモットーに、よりよい品をより安く、迅速にお届けします。

全国通信販売
北海道から沖縄まで

★送料はご注文の際にお問い合わせ下さい。
★掲載の商品は、すべて新品、保証書付きです。
★掲載の商品は充分用意してありますが、ご注文の際は、在庫の確認の上、現金書留または、銀行振込でお申し込み下さい。全商品クレジットでも扱っております。
★お申し込みの際は必ず電話番号を明記して下さい。
★商品、品切れの節はご容赦下さい。

富士銀行八王子支店 （普）1752505

Let's CALL

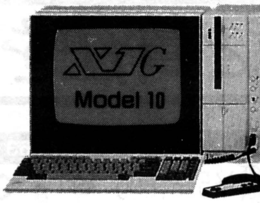
超優良中古パソコンが電話一本で買える!! 03(797)1221



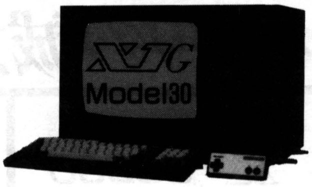
SHARP
CZ-801C (X-1C)
¥119,800➡ ¥10,000



SHARP
CZ-811C (X-1F/10)
¥89,800➡ ¥12,000



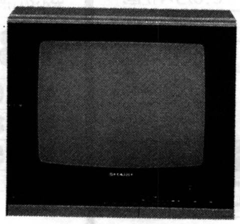
SHARP
CZ-820CE (X-1Gモデル10)
¥69,800➡ ¥16,800
X-1Gモデル10RFコンバータセット
(本体＋AN-58C)
¥72,780➡ ¥19,600
X-1Gモデル10ディスプレイセット
(本体＋CU-14GB)
¥119,600➡ ¥46,600



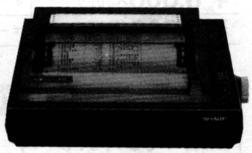
SHARP
CZ-822CB (X1-Gモデル30)
¥118,000➡ ¥59,800
X-1Gモデル30ディスプレイセット
(本体＋CU-14GB)
¥167,800➡ ¥89,600
X-1Gモデル30TVディスプレイセット
(本体＋TVディスプレイ)
¥197,800➡ ¥99,600



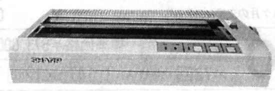
SHARP
CZ-880CB 新品同様
(X-1Turbo Z本体)
¥218,000➡ ¥102,000
CZ-880DB 新品同様
¥109,800➡ ¥86,000
セット価格
¥327,800➡ ¥188,000



SHARP
CZ-820DE・B 新品
(14インチ2000字RGBTV)
¥79,800➡ ¥39,800



SHARP
CZ-8PK5 新品同様
(10インチ漢字プリンタ)
¥129,000➡ ¥69,800
CZ-8PK6 新品同様
(15インチ漢字プリンタ)
¥159,000➡ ¥89,800



SHARP
MZ-1P17(E・B) 新品
(色、グレー・ブラック)
(80桁カラー漢字熱転写プリンタ)
¥76,600➡ ¥42,800
(X1用ケーブル付)
¥76,600➡ ¥46,800
(MZ-2500用ケーブル付)

SHARP

本体

CZ-801C(X-1C)	¥119,800➡	¥ 10,000
CZ-804C(X-1CK)	¥139,000➡	¥ 12,000
CZ-812C(X-1F model 20)	¥139,800➡	¥ 42,000
CZ-822C(X-1G model 30)	¥118,000➡	¥ 52,000
CZ-850C(X-1Turbo/model 10)	¥168,000➡	¥ 25,000
MZ-1500	¥89,800➡	¥ 18,000
MZ-2200	¥128,000➡	¥ 18,000
MZ-2531(MZ-2500V2)	¥198,800➡	¥ 88,000

ディスプレイ

12M-18B(12"グリーン4050文字)	¥44,800➡	¥ 20,000
12M-212C(12"カラー2000文字)	¥99,800➡	¥ 20,000
12M-312C(12"カラー2000文字)	¥89,800➡	¥ 20,000
12M-314C(12"カラー4050文字)	¥128,000➡	¥ 45,000
14M-111C(14"カラー1000文字)	¥67,800➡	¥ 15,000
14M-522C(14"カラー4050文字)	¥99,800➡	¥ 45,000
CU-14F1(14"カラー2000文字)	¥64,800➡	¥ 18,000
CU-14AG2(14"カラー4050文字)	¥84,800➡	¥ 45,000
CZ-801D(14"カラー2000文字RGBTV)	¥99,800➡	¥ 30,000
CZ-850D(15"カラー4050文字RGBTV)	¥129,800➡	¥ 52,000
MZ-1D05(14"カラー2000文字)	¥69,800➡	¥ 18,000
MZ-1D11(12"カラー4000文字)	¥113,000➡	¥ 42,000
MZ-1D15(14"カラー2000文字)	¥72,000➡	¥ 18,000

ディスクドライブ・プリンタ・他

CZ-501F(5"2D、2ドライブ)	¥129,800➡	¥ 55,000
CZ-502F(5"2D、2ドライブ)	¥99,800➡	¥ 55,000
CZ-503F(5"2D、1ドライブ)	¥49,800➡	¥ 32,000
MZ-1F07(5"2D、2ドライブ)	¥158,000➡	¥ 55,000
CZ-800P(10"ドットプリンタ)	¥142,800➡	¥ 15,000
CZ-81P(ミニサイズプリンタ)	¥34,800➡	¥ 10,000
CZ-8PP2(カラープロッタプリンタ) 新品	¥54,800➡	¥ 15,000
CZ-8PD2(10"ドットプリンタ)	¥79,800➡	¥ 28,000
CZ-8PD3(10"ドットプリンタ)	¥59,800➡	¥ 28,000
CZ-8PN1(80桁24ドット漢字熱転写プリンタ)	¥134,800➡	¥ 32,000
MZ-80P6(80桁ドットプリンタ)	¥ ➡	¥ 18,000
MZ-1P06(80桁漢字プリンタ)	¥234,000➡	¥ 45,000
MZ-1P09(MZ-1500用カラープロッタプリンタ) 新品	¥47,600➡	¥ 25,800
MZ-1P17(80桁24ドットカラー漢字熱転写プリンタ・CZ用ケーブル付) 新品	¥76,600➡	¥ 42,800
MZ-1P17(80桁24ドットカラー漢字熱転写プリンタ・MZ用ケーブル付) 新品	¥76,600➡	¥ 46,800
MZ-1T02(MZ-2200専用データレコーダ)	¥19,800➡	¥ 6,000
CZ-81EB(X-1シリーズ用拡張I/Oボックス)	¥29,800➡	¥ 12,000
CZ-8BM2(X-1シリーズ用マウスインターフェイス)	¥19,800➡	¥ 10,000
CZ-8RB(X-1シリーズ用ROM BASIC)	¥19,800➡	¥ 10,000
CZ-8SS2(システムスタンド) 新品同様	¥5,500➡	¥ 4,000

＊SHARP X-1シリーズ特選極上品コーナー＊

CZ-820CE(X-1G/10) 新品同様	¥69,800➡	¥ 16,800
CZ-822CB(X-1G/30) 新品同様	¥118,000➡	¥ 59,800
CZ-880CB(X-1TurboZ) 新品同様	¥218,000➡	¥102,000

＊SHARP ディスプレイ特選極上品コーナー＊

MD-12P1(12"グリーン4050文字) 新品同様	¥39,800➡	¥ 29,800
CU-14G(14"カラー2000文字) 新品	¥49,800➡	¥ 29,800
CU-14A4(14"カラー4050文字) 新品	¥89,800➡	¥ 49,800
CZ-820D(14"カラー2000文字RGBTV) 新品同様	¥79,800➡	¥ 39,800
CZ-880DB(15"カラー4050文字RGBTV) 新品同様	¥109,800➡	¥ 86,000
CZ-600D(15"カラー4050文字RGBTV) 新品同様	¥129,800➡	¥ 88,000

●電話一本で高額下取り、即商品はお手元へ!
●あなたの不要になったパソコンを電話一本で査定し買取ります。
●掲載の商品以外も取り扱っておりますのでお気軽にお電話下さい。

▼本社注文デスク
03(797)1221

コンピュータバンク

株式会社パシフィックコンピュータバンク　〒150 東京都渋谷区渋谷1-6-8 井上ビル　営業時間／AM9:30〜PM9:30 年中無休

全商品保証付 6ヶ月の保証期間だから安心です。
クレジットでOK カレッジクレジットも取扱います。
全国無料配送 全国どこでも配達料はいただきません。
日曜配達可 留守の多い方でも安心です。
高額下取り 少ない予算で買いかえもラクラク。
高額買取り 電話1本で即、現金お支払い。
代金引換えシステム 商品到着時の代金支払いでOK。
ボーナス一括払い 商品は即お手元へ、お支払いはボーナス時に。

できうる限りの誠意と責任をもって———IPL。

SHARP X-68000

アクセス No.X0760

価格¥573,000 ➡ 超特価 **CALL!!**

CZ-600C(65536同時発色、スーパーインポーズ、ステレオFM音源)	¥369,000
CZ-600D(4096色TV19モード多機能リモコン付)	¥129,800
CZ-213MS(MUSIC PRO 68K)	¥ 18,800
CZ-214MS(SOUND PRO 68K)	¥ 15,800
CZ-217AS(ツインビーシューティングゲーム)	¥ 7,800
CZ-222AS(アルカノド・リベンジ・オブ・ドー(ブロックゲーム))	¥ 7,800
3Mブランクディスケット(5"2HD＊10枚)	¥ 24,000
CZ-8NJ1(ジョイカード プレゼント!)	¥ 0
安心の3倍保証(IPL保証書付き)	¥ 0
初期不良期間(ワイドに1ケ月の交換システム!)	¥ 0

¥1,000 ×72回 ボーナス 3.14万×12回 標準価格¥573,000

¥ 2,900×72回	ボーナス	2.0万×12回
¥ 4,000×48回	ボーナス	2.75万×8回
¥ 6,100×36回	ボーナス	3.0万×6回

アクセス No.X0761

価格¥545,200 ➡ 超特価 **CALL!!**

CZ-600C(65536同時発色、スーパーインポーズ、ステレオFM音源)	¥369,000
CZ-600D(4096色TV19モード多機能リモコン付)	¥129,800
源平討魔伝	¥ 7,800
スペースハリアー	¥ 6,800
マンハッタンレクイエム	¥ 7,800
3Mブランクディクケット(5"2HD＊10枚)	¥ 24,000
CZ-8NJ1(ジョイカードプレゼント!)	¥ 0
初期不良期間(ワイドに1ケ月の交換システム!)	¥ 0
安心の3倍保証(IPL保証書付き)	¥ 0

¥1,000 ×72回 ボーナス 2.89万×12回 標準価格¥545,200

¥ 3,000×48回	ボーナス	3.0万×8回
¥ 5,000×36回	ボーナス	3.18万×6回
¥ 10,100×24回	ボーナス	3.0万×4回

Level Up

下取りもご相談ください。

SHARP X68000 ACE HD

アクセス No.X0762

価格¥742,670 ➡ 超特価 **CALL!!**

CZ-611C(20MHDD搭載、65536色発色、FM8音源内蔵)	¥399,800
CZ-611D(0.31ミリ、アナログ3モードオートスキャン)	¥145,000
CZ-6ST1(角度自由自在、調節OK!)	¥ 5,800
CZ-211LS(C compileソフト開発を効率良くサポート)	¥ 39,800
Z's STAFF PRO 68K(グラフイックツール)	¥ 58,000
CZ-8PC2(10"カラー熱転写B5～B4ハガキ可、全角半角文字)	¥ 69,800
3Mブランクディスケット(5"2HD 10枚)	¥ 24,000
A4カット紙(100枚)	¥ 470
電話帳電卓プレゼント(電話番号50人分、スケジュールメモOK電卓機能付)	¥ 0
初期不良期間(ワイドに1ケ月間交換システム!)	¥ 0
安心の3倍保証(IPL保証書付き)	¥ 0

¥3,000 ×72回 ボーナス 4.05万×12回 標準価格¥742,670

¥ 5,000×72回	ボーナス	2.85万×12回
¥ 8,000×48回	ボーナス	3.33万×8回
¥ 9,200×36回	ボーナス	5.0万×6回

アクセス No.X0763

価格¥576,600 ➡ 超特価 **CALL!!**

CZ-611C(20MHDD搭載、65536色発色、FM8音源内蔵)	¥399,800
CZ-611D(0.31ミリ、アナログ3モードオートスキャン)	¥145,000
3Mブランクディスケット(5"2HD＊10枚)	¥ 24,000
源平討魔伝	¥ 7,800
CZ-8NJ1(ジョイカードプレゼント!)	¥ 0
初期不良期間(ワイドに1ケ月の交換システム!)	¥ 0
安心の3倍保証(IPL保証書付き)	¥ 0

¥2,400 ×72回 ボーナス 3.0万×12回 標準価格¥576,600

¥ 5,000×48回	ボーナス	3.13万×8回
¥ 8,000×36回	ボーナス	3.12万×6回
¥10,900×24回	ボーナス	5.0万×4回

アクセス No.X0764

価格¥954,670 ➡ 超特価 **CALL!!**

CZ-611C(20MHDD搭載、65536色発色、FM8音源内蔵)	¥399,800
CZ-611D(0.31ミリ、アナログ3モードオートスキャン)	¥145,000
CZ-211LS(C compileソフト開発を効率良くサポート)	¥ 39,800
Z's STAFF PRO 68K(グラフイックツール)	¥ 58,000
CZ-8NSI(フルカラーA4ズーム機能色ずれの少ない線順次方式ソフト付き■)	¥188,000
CZ-6BN1(68000用スキャナ用パラレルボード)	¥ 29,800
CZ-8PC2(10"カラー熱転写B5～B4ハガキ可、全角半角文字)	¥ 69,800
3Mブランクディスケット(5"2HD＊10枚)	¥ 24,000
A4カット紙(100枚)	¥ 470
電話帳電卓プレゼント(電話番号50人分、スケジュールメモOK電卓機能付)	¥ 0
初期不良期間(ワイドに1ケ月間交換システム!)	¥ 0
安心の3倍保証(IPL保証書付き)	¥ 0

¥4,300 ×72回 ボーナス 5.0万×12回 標準価格¥954,670

¥ 8,000×72回	ボーナス	2.75万×12回
¥ 9,200×48回	ボーナス	5.0万×8回
¥14,200×36回	ボーナス	5.0万×6回

アクセス No.X0766

価格¥836,500 ➡ 超特価 **CALL!!**

CZ-611C(20MHDD搭載、65536色発色、FM8音源内蔵)	¥399,800
CZ-611D(0.31ミリ、アナログ3モードオートスキャン)	¥145,000
CZ-6ST1(角度自由自在、調節OK!)	¥ 5,800
CZ-211LS(C compileソフト開発を効率良くサポート)	¥ 39,800
CZ-212BS(データベース表グラフ、ソート機能、斜線、横倍角、網掛け下線)	¥ 68,000
CZ-8PK8(15"ハガキ封筒可、用紙をずらさないトラクタフィーダ付き)	¥152,000
3Mブランクディスケット(5"2HD＊10枚)	¥ 24,000
ペーパー15インチ(500枚)	¥ 2,100
電話帳電卓プレゼント(電話番号50人分、スケジュールメモOK電卓機能付)	¥ 0
初期不良期間(ワイドに1ケ月間交換システム!)	¥ 0
安心の3倍保証(IPL保証書付き)	¥ 0

¥2,600 ×72回 ボーナス 5.0万×12回 標準価格¥836,500

¥ 5,000×72回	ボーナス	3.53万×12回
¥10,000×48回	ボーナス	3.1万×8回
¥ 11,200×36回	ボーナス	5.0万×6回

IPL TOPICS

日本テレビ系火曜サスペンス劇場「ハネムーン」/テレビ朝日土曜ワイド劇場「黒い仮面の美女」。「日時計館の美女」又、フジテレビ系列、月曜ドラマランドなど他多数の番組で使用するコンピュータプログラムをIPLが制作。

アクセス No.X0765

価格¥877,400 ➡ 超特価 **CALL!!**

CZ-611C(20MHDD搭載、65536色発色、FM8音源内蔵)	¥399,800
CZ-611D(31ミリ、アナログ3モードオートスュャン)	¥145,000
CZ-6ST1(角度自由自在、調節OK!)	¥ 5,800
CZ-6BE1A(1MB増設RAMボード)	¥ 35,000
CZ-6VT1(カラーイメージユニット、テロッパー機能付き)	¥ 69,800
CZ-6PV1(カラービデオプリンタ)	¥198,000
3Mブランクディスケット(5"2HD＊10枚)	¥ 24,000
電話帳電卓プレゼント(電話番号50人分、スケジュールメモOK電卓機能付)	¥ 0
初期不良期間(ワイドに1ケ月間交換システム!)	¥ 0
安心の3倍保証(IPL保証書付き)	¥ 0

¥3,300 ×72回 ボーナス 5.0万×12回 標準価格¥877,400

¥ 6,600×72回	ボーナス	3.0万×12回
¥ 7,800×48回	ボーナス	5.0万×8回
¥12,500×36回	ボーナス	5.0万×6回

SHARP X1 twin

アクセス No.X0767

価格¥215,800 ➡ 超特価 **CALL!!**

CZ-830CBK(X1twin)	¥ 99,800
CZ-830D(14"カラー、ビデオ入力端子付きRGB、A/D対応在)	¥ 98,000
3Mブランクディスケット(3.5"2DD＊10枚)	¥ 18,000

¥1,000 ×72回 ボーナス 1.05万×12回 標準価格¥215,800

¥ 3,800×48回	ボーナス	なし
¥ 4,900×36回	ボーナス	なし

7 お支払い ヶ月先からOK

翌月一括から自由に設定

(¥) 月々わずか1000円

SHARP X1 turbo Z

アクセス No.X0768

価格¥335,000 ➡ 超特価 **CALL!!**

CZ-881C(ビデオ画像取込、スーパーインポーズ、テロップ可、マウス付)	¥179,800
CZ-880D(14"カラーTVリモコン付4050文字)	¥109,800
3Mブランクディスケット(5"2HD＊10枚)	¥ 24,000
イースII	¥ 7,800
紫醜羅	¥ 7,800
ザナドゥ シナリオ2	¥ 5,800
初期不良期間(ワイドに1ケ月の交換システム!)	¥ 0
安心の3倍保証(IPL保証書付き)	¥ 0

¥1,000 ×72回 ボーナス 1.92万×12回 標準価格¥335,000

¥ 3,000×36回	ボーナス	2.66万×6回
¥ 5,000×24回	ボーナス	3.52万×4回
¥ 4,800×60回	ボーナス	なし

超 低 金 利 ……… 組 合 せ 自 由 全 国 無 料 配 送

※今回掲載の製品は、6月18日より7月18日までの期間に限らせていただきます。

COMPUTER CREATION

IPL

株式会社・アイピーエル

〒248 鎌倉市雪ノ下4−1−12 雪ノ下ビル

鎌倉市雪ノ下3−4−23：商品管理部

AM10:00▶PM8:00

水曜日定休

安心の3倍保証

50,000人もの人々が体感した安心感。
信頼のIPLワンタッチワイドサポート

●業界初、IPLでこそ成し得た3倍保証。

メーカー保証12ヶ月の商品なら36ヶ月の保証とグッと長期間の保証を実施。末長く安心してご利用いただけるよう、IPLが成し得たワイドなサポート体制。

●IPLだからこそ初期不良への保証も万全。交換期間も1ヶ月ともっとも長期間です。

プリンタヘッド交換￥29,500以上／98シリーズメインボード交換￥21,600以上／ドライブ交換￥13,200以上

INTELGENT POWER UP
IPLニューショールームオープン
——知的フィールドを実感——

比べてほしいから、ご紹介します。
さらにお買得IPLクレジット

●ステップアップクレジットがおトク。

まず月々1,000円からスタートして2年後から3,000円へアップ。ボーナスも1年後1万円。3年後3万円。また夏のボーナスを貯金して冬のボーナスも1年後1万円。3年後3万円、また夏のボーナスを貯金して冬のボーナスのみ年一回のお支払いもOK。さらにお支払い回数も1回払いから最長72回までご自由に設定が可能です。

●追加購入もクレジットだから便利。

追加購入も買い換えもご利用中のIPLクレジットを月々僅か1,000円ずつの調整でOK。

●プラスαフェアを実施中!! お買得感をじっくり比べて下さい。

Order Telephone

●本社 0467-24-7511　●大阪 06-311-2736

●銀座 03-541-3058　●青山 03-470-0061　●札幌 011-621-1444

●仙台 022-266-0531　●広島 082-293-7881　●福岡 092-481-2644

●商品管理部 納期、配達日のお問合せ、ご指定日のご連絡 0467-24-1154/●ご注文お問合せ 0467-24-1154/●メンテナンス部 ハード上のご相談、お問合せ、初期不良の対応 0467-24-0453

●FAX ご注文、お見積り、カタログ編集などスピーディに 0467-24-0561/●タイムリーボックス ホットな新製品ニュースをお知らせします。0467-24-0941/●下取りホットライン 0467-24-2040

本社：〒248 鎌倉市雪ノ下4-1-12 雪ノ下ビル　電話受付：AM10：00〜PM8：00 水曜日定休

商品管理部：〒248 鎌倉市雪ノ下3-4-2　電話受付：AM10：00〜PM8：00 水曜日定休

SHARP X68000 ACE

アクセス **No.X0769**

価格￥469,400 ➡ 超特価 **CALL!!**

CZ-601C（CPU6800, 2Mバイト、65536同時発色）	￥319,800
CZ-601D（0.39ミリ、アナログ3モードオートスキャン）	￥119,800
CZ-6ST1（角度自由自在、調節OK!）	￥5,800
3Mブランクディスケット（5"2HD*10枚）	￥24,000
CZ-8NJ1（ジョイカードプレゼント!）	￥0
初期不良期間（ワイドに1ケ月間交換システム!）	￥0
安心の3倍保証（IPL保証書付き）	￥0

標準価格￥469,400

￥1,000 ×72回 ボーナス 2.98万×12回

￥5,000×36回	ボーナス	3.35万×6回
￥10,000×24回	ボーナス	3.25万×4回
￥4,900×48回	ボーナス	2.0万×8回

50,000人もの人々が体感した安心感。
——信頼のIPLワイドサポート

アクセス **No.X0770**

価格￥637,470 ➡ 超特価 **CALL!!**

CZ-601C（CPU68000, 2Mバイト、65536同時発色）	￥319,800
CZ-601D（, 39ミリ、アナログ3モードオートスキャン）	￥119,800
CZ-6ST1（角度自由自在、調節OK!）	￥5,800
CZ-211LS（C compileソフト開発を効率良くサポート）	￥39,800
Z's STAFF PRO 68K（グラフィックツール）	￥58,000
CZ-8PC2（10"カラー熱転写B5〜B4ハガキ可、全角半角文字）	￥69,800
3Mブランクディスケット（5"2HD*10枚）	￥24,000
A4カット紙（100枚）	￥470
電話帳電卓プレゼント（電話番号50人分、スケジュールメモOK電卓機能付）	￥0
初期不良期間（ワイドに1ケ月間交換システム!）	￥0
安心の3倍保証（IPL保証書付き）	￥0

標準価格￥637,470

￥3,000 ×72回 ボーナス 3.2万×12回

￥5,000×72回	ボーナス	2.0万×12回
￥8,000×48回	ボーナス	2.15万×8回
￥10,000×36回	ボーナス	3.0万×6回

組み合せ自由



プラスαフェア

実施6/18(SAT)〜7/18(MON)

Chance 1 期間中、システムお買い上げの方、先着200名様に、電話帳電卓をプレゼント（電話番号・スケジュールを記憶、10桁電卓機能付）

Chance 2 期間中、テスクをお買上げの方全員に、A-300（原稿用スタント￥8,000）をプレゼント

Chance 3 期間中、シャープ製品をシステムでお買上げの方全員にCZ-8NJ1（ジョイカード）をフレゼント

GOOD CHOICE CORNER

IPL厳選／IPLワイドサポートOK！ あなたらし差の組み合せを応援します。

SHARP

①CZ-502F（ミニフロッピーディスクユニット(2D)）	￥98,000➡￥48,000
②CZ-82FR（増設用フロッピーディスクユニット／CZ-802C用）	新品￥59,800➡￥28,000
③CZ-8BV1（カラーイメージボード）	新品￥39,800➡￥18,000
④CZ-8EB3（カクチョウⅠ／0ボックス／X1シリーズ、X1ターボシリーズ）	￥33,000➡￥18,000
⑤CZ-8KR（カンジROM／X1, X1C, X1CS, X1Dヨウ）	￥38,000➡￥8,000
⑥CZ-8NM1（X1 turbo用 マウス）	￥13,800➡￥6,000
⑦MZ-1M08（ボイスボード）	￥10,000➡￥6,000
⑧MZ-8B104（MZ-2200・GP-IBインタフェイスカード）	新品￥45,000➡￥18,000
⑨MZ-2521（CPU MZ-2521）	￥198,000➡￥52,000
⑩PC-5,000（CPU）	￥350,000➡￥198,000
⑪X1 turbo Ⅲ セット（CZ-870CB+CZ-870DB）	￥277,800➡￥148,000
⑫X1F model10セット（CZ-811CE+CZ-811DE）	￥179,600➡￥48,000
⑬X1F model10（CZ-811CR）	新品￥89,800➡￥16,000
⑭CU-12P1（12インチ0.28ドットRGBカラーCRT）	新品￥118,000➡￥48,000
⑮MZ-1P17（10インチ24ドットカラー漢字サーマルプリンター）	￥79,800➡￥32,000
⑯MZ-1R05（MZ-3500シリーズ用漢字メモリー）	新品￥38,000➡￥8,000
⑰CZ-112SF（X1用 NEW BASIC(V2.0)カセット版）	新品￥7,800➡￥3,000
⑱CZ-122PF（THE YOKOZUNA カセット版）	新品￥15,800➡￥6,000

輸送上のトラブルにも対応　お申し込みはナンバーでお願いします。

Jun.18〜July.18

J&P オプション通信販売

J-DMA 社団法人日本通信販売協会 正会員店

全国どこでも無料配達

パソコン通信

J&P HOT LINE でもお申し込みいただけます。

J&Pメールショッ

■シンプルで使いやすいパソコンラック・デスク・チェアー

X7-1
パソコンラック&チェアーセット
ラック寸法 幅600mm3段棚
ラック：エレコムDS-10
チェアー：コイズミL-395
メーカー標準価格合計44,000円
セット特価 **23,000円**
●シートカラー ①青色 ②茶色

X7-2
パソコンシステムデスク
エレコム ER-1200
J&P特価**29,000円**
幅1200×高さ650～1180 奥行750mm

X7-3
エレコム
PD-02
メーカー標準価格43,000円
J&P特価**19,800円**
コード落とし付
幅640%×高さ1305%×奥行700%

X7-4
エレコム
PD-99+FO-60E
セット
メーカー標準価格合計51,500円
J&P特価**33,000円**
トレーユニット（FO-60E）をセットしてお得。
幅900%×高さ1280%×奥行700%

X7-5
パソコンチェアー
コイズミL-395
キャスター付
メーカー標準価格12,000円
J&P特価**6,800円**
シートカラー ①青色 ②茶色

■パソコングッズ

X7-6
OA電源タップ
ナショナルWCH 4511
ノイズフィルター 集中スイッチ付
J&P特価**6,980円**

X7-7
TVフィルター（14インチ用）
東レEフィルターNEW14
J&P特価**9,600円**

X7-8
エレコムSO-450
J&P特価**3,300円**
原稿が見やすく場所をとりません。

X7-9
5インチケース
100枚収納可
J&P特価**2,000円**

X7-10
3.5インチケース
80枚収納可
J&P特価**2,000円**

X7-11
PS-80
10インチプリンタスタンド
J&P価格**3,400円**
※プリンタ別売

X7-12
MS-300
J&P特価**3,500円**
ディスプレイの角度を自由に調整できます。

■各種切替器

X7-13
パソコン切替器
J&P価格**9,800円**
パソコン1・パソコン2―プリンタ
KSW C
1台のプリンタと2台のパソコンを切替えます。

X7-14
ディスプレイ切替器
J&P価格**9,800円**
パソコン1・パソコン2―カラー・グリーン
KSW D　8ピンRGB、グリーン端子付

X7-15
モデム、RS232C 切替器
J&P価格**12,800円**
パソコン―モデム1・モデム2
KSW M
1台のパソコンで2台のRS-232C機器が使えます。

X7-16
X-1プリンタ切替器
J&P価格 **12,800円**
X-1―プリンタ1・プリンタ2
KSW-X1
X-1で2台のプリンタを切替えて使えます。

■電子手帳

シャープPA-7000
J&P特価**17,800円**
これ1台で、電卓・電話帳・スケジュール・メモ・カレンダー機能があります。別売のモジュールを使うことにより、漢字辞書や英和・和英の翻訳機としても使えます。学生、技術者からビジネスマンまで幅広くお使いいただけます。

X7-17

X7-18
ICカード（PA-7000用）

❶PA-7C1	英和・和英カード	6,300円
❷PA-7C2	漢字辞書カード	9,000円
❸PA-7C3	6ケ国語会話カード	6,300円
❹PA-7C4	カラオケ歌詞カード	9,000円
❺PA-7C10	電話帳・住所録カード	9,000円
❻PA-7C11	販売管理カード	9,000円
❼PA-7C12	技術計算カード	6,300円

X7-19
周辺機器

❶CE-152	データレコーダ	9,800円
❷CE-50P	プリンタ	17,800円
❸CE-200L	通信用ケーブル	2,500円

■ポケコン

X7-20
PC-E200 J&P特価**17,800円**
Z80CPU採用で高速演算を実現。24桁4行表示

X7-21
PC-E500 J&P特価**24,800円**
充実の124関数機能、最大96Kバイトまで増設可能。40桁4行表示

さあ始めようパソコン通信

■X-1通信セット

X7-24
モデム：CZ-8TM2
J&P HOTLINE：スタータキット
通信速度300・1200bps
標準価格合計52,800円
セット価格**49,800円**

■X-1ターボ通信セット

X7-25
モデム：アイワ PV-A1200MKⅡ
通信ソフト：SPS JETターボターミナル
J&P HOTLINE：スタータキット
通信速度300・1200bps
標準価格合計39,600円 セット価格**39,600円**

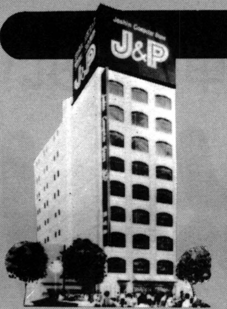
■プリンタ

X7-22
10インチワイヤドット
CZ-8PK9
ハガキ可
J&P価格**89,800円**
X-1・X-68000用

X7-23
シャープ CZ-8PC2
J&P価格**69,800円**
10インチカラー熱転写
X-1・X-68000用

全国無料配達

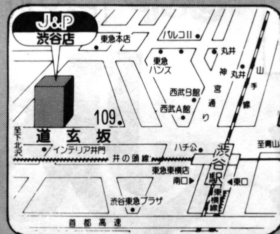
ピング



メールショッピングのお申し込みは J&P 渋谷店で承ります。

フロアーごあんない

- 4F パソコン教室
  - ●パソコン入門コース ●BASIC上級コース ●BASIC初級コース ●各種ビジネスコース
- 3F OA機器
  - ●ビジネスパソコン ●ワードプロセッサ ●ビジネスソフト ●OAサプライ ●ハンドヘルドコンピュータ
- 2F ビジネスパソコン
  - ●パソコン ●ディスプレイ ●プリンタ ●専門書籍 ●パソコンアクセサリー
- 1F ホビーのパソコン
  - ●ホビーパソコン ●MSX ●ゲームソフト ●学習ソフト



Personal Computer Store
J&P
渋谷店

東京都渋谷区道玄坂2丁目28番4号(〒150)
☎(03)496-4141
定休：毎週水曜日



X7-26

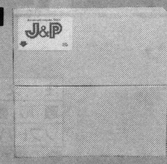
■ディスケット

マクセル

❶MD2-D(10枚)	1,900円
❷MD2-DD(10枚)	2,400円
❸MD2-256HD(10枚)	2,500円
❹MF1-D(10枚)	4,500円
❺MF2-D(10枚)	4,500円
❻MF1-DD(10枚)	4,500円
❼MF2-DD(10枚)	5,000円
❽MF2-256HD(10枚)	8,300円

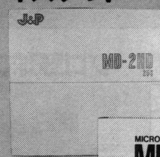
■J&Pオリジナルディスケット

X7-27



J&Pオリジナル
MD-2D(20枚)
3,000円

X7-28



MD-2HD(10枚) 2,100円

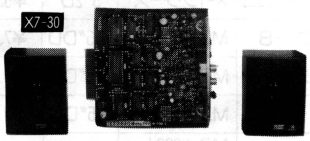
X7-29



MF-2DD(10枚)
4,000円

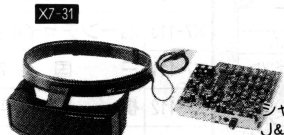
■〈X-1/ターボオプション〉

X7-30



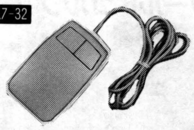
FM音源ボード
シャープCZ-8BS1 J&P価格23,800円
X-1用8重和音200音色、ステレオ サウンドのFM音源

X7-31



立体映像セット
シャープCZ-8BR1
J&P価格29,800円
X-1/X-1ターボシリーズにて立体映像が楽しめます。
立体作画ソフト・立体スコープ付

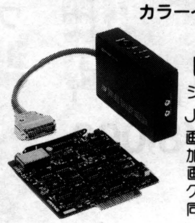
X7-32



マウス
シャープCZ-8NM2
J&P価格6,800円
X-1・MZ用マウス

カラーイメージボード

X7-33



シャープCZ-8BV2
J&P価格39,800円
画像を自在に修正・加工できます
画像処理ツール・グラフィックソフト同梱

■X68000オプション X7-35

❶CZ-6BE1	1MB増設メモリ	35,000円
❷CZ-6BE2	2MB増設メモリ	79,800円
❸CZ-6BE4	4MB増設メモリ	138,000円
❹CZ-6BU1	ユニバーサルI/Oボード	39,800円
❺CZ-6BG1	GP-IBボード	59,800円
❻CZ-6BF1	RS-232C増設2チャンネル	49,800円
❼CZ-6BP1	68881数値演算プロセッサボード	79,800円
❽CZ-6EB1	拡張I/Oボックス4スロット	88,000円

■MZ-2500システムソフト X7-36

商品名	機種名	価格
FORTRAN	❶IP-1213	13,800円
C言語	❷IP-1214	13,800円
COBOL	❸IP-1215	13,800円
LISP	❹IP-1216	13,800円
PROLOG	❺IP-1217	13,800円
CPM	❻MZ-6Z001	16,800円

■プリンタオプション X7-37

❶MZ-1C48	X-1シリーズ用プリンタケーブル	6,800円
❷MZ-1C35	MZ-2500/2200/2000用ケーブル	6,800円
❸MZ-1R29	MZ-1P17(B)用第2水準ROM	14,800円
❹CZ-8PC1-3	CZ-8PC1用第2水準ROM	9,800円

■X-1/X-1ターボシステムソフト X7-38

商品名		型番	価格
ランゲージマスター(CP/M®)		❶CZ-128SF(2D・5″FD版)	9,800円
turbo CP/M(漢字版)		❷CZ-130SF(2D・5″FD版)	14,800円
X-1 LOGO		❸CZ-134SF(2D・5″FD版)	9,800円
turbo Z's STAFF		❹CZ-137SF(2D・5″FD版)	19,800円
X1 Z's STAFF		❺CZ-138SF(2D・5″FD版)	13,800円
ミュートピア		❻CZ-139SF(2D・5″FD版)	12,800円
グラフィックライブラリー		❼CZ-140SF(2D・5″FD版)	9,800円
NEW Z-BASIC		❽CZ-141SF(2HD・5″FD版)	18,800円
		❾CZ-141SF(2D・5″FD版)	18,800円
ランゲージシリーズ	FORTRAN	❿CZ-115LF(2D・5″FD版)	13,800円
	C	⓫CZ-116LF(2D・5″FD版)	13,800円
	turbo LOGO(漢字版)	⓬CZ-117SF(2D・5″FD版)	18,800円
	COBOL	⓭CZ-118LF(2D・5″FD版)	13,800円
	PROLOG	⓮CZ-119LF(2D・5″FD版)	13,800円
	LISP	⓯CZ-120LF(2D・5″FD版)	13,800円
	APL	⓰CZ-126LF(2D・5″FD版)	13,800円
X-1 NEW BASIC		⓱CZ-112SF(カセット版)	7,800円
		⓲CZ-113SF(3″FD版)	8,800円
		⓳CZ-124SF(2D・5″FD版)	8,800円

X7-34

❶CZ-8BE2
J&P価格29,800円
320KB外部メモリ

❷CZ-8BM2
J&P価格19,800円
RS-232C・マウスボードX-1用

❸CZ-8EP
J&P価格11,800円
拡張IOポート(4口)X-1用

❹CZ-8EB3
J&P33,800円
拡張IOボックス(4スロット)X-1用

■各種漢字ROM X7-39

❶CZ-8BK2	X-1F第1水準ROM	19,800円
❷CZ-8BK3	X-1ターボ第2水準ROM	13,800円
❸CZ-8BK4	X-1ターボ2第2水準ROM	6,800円

お申し込み方法

右の注文書にご希望商品の注文No.および必要事項ご記入の上、現金書留にて J&P 渋谷店までお申し込みください。現金受領後、発送また、J&P HOTLINE会員の方は、ショッピングコーナーでもお申し込みいただけます。

●記載以外のご注文も承りますので、詳しくはお電話にてお問い合わせ下さい。

☎(03)496-4141
定休：毎週水曜日

キリトリ線

現金書留申込み用紙

	注文No	数量	金額
おところ 〒□□□-□□	X7- ( )		円
	X7- ( )		円
TEL ( )	合計		円
おなまえ 様	お手持ちのパソコン		

お申込み先：東京都渋谷区道玄坂2丁目28番4号(〒150) J&P 渋谷店メールショッピング係

J&P ソフト通信販売

社団法人日本通信販売協会 正会員店

全国どこでも
無料配達



パソコン通信

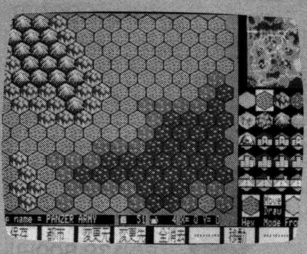
J&P HOT LINE でもお申し込みいただけます。

J&P メールショッ

■ビックヒットソフト

スーパー大戦略

注文No.	X7-100
適応機種	X-1ターボ
ソフトハウス	システムソフト



5″2D ¥8,000

プロ野球ファン

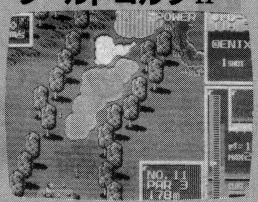


¥7,800(5″2D)

注文No.	X7-101
適応機種	X-1シリーズ
ソフトハウス	日本テレネット

春の高校野球がスタートするまで冬眠でもしようと考えていた野球ファンのあなた。さあ、この真新しい球場で白球に賭けた男たちのドラマを味わってみてください。

ワールドゴルフII



¥7,800(5″2D)

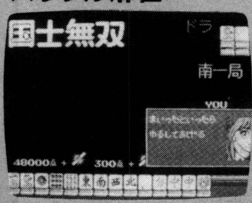
注文No.	X7-102
適応機種	X-1ターボ
ソフトハウス	日本テレネット

2モード(トレーニングモード、トーナメントモード)全72ホール構成。トーナメントモードでは、計100人のライバルゴルファーが登場。あらゆる角度からゴルフのおもしろさを徹底的に分析し、それをシミュレート。ゴルフゲームの最高峰作品!

注文No.	タイトル	ソフトハウス	適応機種	メディア	価格
X7-103	レリクス	ボーステック	X-1シリーズ	5″2D	¥7,500
X7-104	信長の野望(全国版)	光栄	X-1シリーズ	5″2D	¥9,800
X7-105	アルバトロス	日本テレネット	X-1シリーズ	5″2D	¥8,800
X7-106	殺意の接吻	リバーヒルソフト	X-1シリーズ X68000	5″2D	¥5,800
X7-107	棋太平	S・P・S	X-1シリーズ	5″2D	¥6,500
X7-108	イース	日本ファルコム	X-1シリーズ	5″2D	¥7,800
X7-109	ザナドゥ・シナリオII	日本ファルコム	X-1シリーズ	5″2D	¥5,800
X7-110	ムーンチャイルド	HOT-B	MZ-2500	3.5″DD	¥7,800
X7-111	三国志	光栄	MZ-2500	3.5″DD	¥14,800
X7-112	棋太平	S・P・S	MZ-2500	3.5″DD	¥7,000
X7-113	ハイドライドII	T&Eソフト	MZ-2000/2200	5″2D	¥6,800
X7-114	レリクス	でんぱ	X68000	5HD	¥7,200

■Xホビーソフト

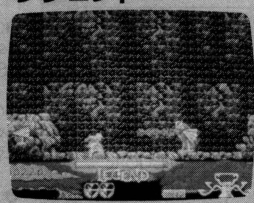
パワフル麻雀



¥6,800(5″2D)

注文No.	X7-115
適応機種	X-1ターボ
ソフトハウス	dBソフト

レジェンド



¥7,800(5″2D)

注文No.	X7-116
適応機種	X-1シリーズ
ソフトハウス	クエイザーソフト

人の心の光と闇を司るクリスタルを妖精アリーナが誤って地上に落してしまった。そのクリスタルを手に入れたのは古しえの時代に神々をも滅ぼそうとした大魔王ガウディアであった。

蒼き狼と白き牝鹿ジンギスカン

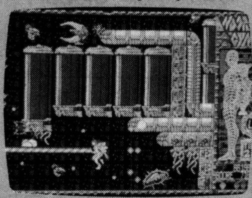


¥9,800(3.5″DD)

注文No.	X7-117
適応機種	MZ-2500
ソフトハウス	光栄

「蒼き狼と白き牝鹿」の壮大なストーリーに加え、戦闘モードでは騎馬隊や弓矢隊など新しく加えられた戦闘部隊や略奪、狩猟、降伏勧告などの新コマンドも加わって、より複雑な戦略が楽しめるシミュレーションゲームとして期待できる。

反生命戦機アンドロギュヌス



¥7,800(5″2D)

注文No.	X7-118
適応機種	X-1シリーズ
ソフトハウス	日本テレネット

"アンドロギュヌス"その名は「両性具有」を意味する。だがおまえはいったい何者なのだ?「人間?それとも機械?」「女性?それとも男性?」「悪漢?それとも救世主?」唯一はっきりしていることは、お前に与えられた使命「宇宙転覆ヲ企ム半物質星"ウルド"を破壊セヨ」。"アンドロギュヌス"よ、お前はまだ自分が何者であるか知らない。

Might and Magic



¥9,800(5″2D)

注文No.	X7-119
適応機種	X1ターボ
ソフトハウス	スタークラフト

薄暗いダンジョンから足を一歩踏み出すと、そこにはまっすぐに延びた並木道があった。我がパーティーの前途には、「バーン」という名のみ知られている未知なる世界が広がっている。かくして旅は始まる。

ジーザス



¥7,800(5″2D)

注文No.	X7-120
適応機種	X1ターボ
ソフトハウス	エニックス

ハレー彗星が接近しつつあった西暦2061年、人類がハレー彗星調査のために飛ばした2機の有人探査機を舞台に、これまでに経験できなかったような感動のドラマを、いまパソコンのAGVとして体験できるときがやってきた。

注文No.	タイトル	ソフトハウス	適応機種	メディア	価格
X7-121	ウィザードリー3	アスキー	X-1ターボ	5″2D	¥9,800
X7-122	サジリ	日本テレネット	X-1ターボ	5″2D	¥7,800
X7-123	魔界復活	ソフトWING	X-1ターボ	5″2D	¥7,800
X7-124	ダ・ビンチ	HAL研究所	X1シリーズ	5″2D	¥6,800
X7-125	ディーヴァ	T&E	X1シリーズ	5″2D	¥7,800
X7-126	ウルティマIV	ポニー	X-1シリーズ	5″2D	¥9,800
X7-127	蒼き狼と白き牝鹿 ジンギスカン	栄光	X-1シリーズ	5″2D	¥9,800
X7-128	スーパーレイドック	T&E	X-1シリーズ	5″2D	¥6,800
X7-129	ドーム	システムサコム	X-1シリーズ	5″2D	¥9,800
X7-130	ガイフレーム	NCS	X-1シリーズ	5″2D	¥7,800
X7-131	抜忍伝説	ブレイングレイ	X-1シリーズ	5″2D	¥9,800
X7-132	ドラゴンバスター	デンパ	X-1シリーズ	5″2D	¥6,200
X7-133	ラビリンス	日本AVC	X-1シリーズ	5″D	¥7,800
X7-134	夢幻戦士ヴァリス	日本テレネット	X-1シリーズ	5″2D	¥7,800
X7-135	ストーム	マイクロネット	X-1シリーズ	5″2D	¥7,800
X7-136	プロフェッショナル麻雀	シャノアール	X-1シリーズ	テープ	¥4,800
X7-137	ガルフォース	スキップトラスト	X-1シリーズ	5″D	¥7,800
X7-138	カーマイン	マイクロキャビン	X-1シリーズ	5″2D	¥7,800
X7-139	麻雀狂時代	マイクロネット	X-1シリーズ	5″2D	¥6,800
X7-140	女神転生	日本テレネット	X-1シリーズ	5″2D	¥7,800
X7-141	上海	システムソフト	X-1シリーズ	5″2D	¥6,500
X7-142	うる星やつら	マイクロキャビン	X-1シリーズ	5″2D	¥6,800
X7-143	ギャンブラー自己中心派	ゲームアーツ	X-1シリーズ	5″2D	¥6,800
X7-144	めぞん一刻	マイクロキャビン	X-1シリーズ	5″2D	¥6,800
X7-145	九玉伝	テクノソフト	MZ-2500	3.5″DD	¥7,800

※㊔は新作ソフトの略です。

全国無料配達

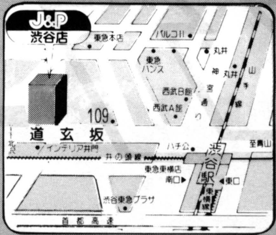
ピング



メールショッピングのお申し込みは J&P 渋谷店で承ります。

フロアーごあんない

- 4F パソコン教室
  - ●パソコン入門コース●BASIC上級コース●BASIC初級コース●各種ビジネスコース
- 3F OA機器
  - ●ビジネスパソコン●ワードプロセッサ●ビジネスソフト●OAサプライ●ハンドヘルドコンピュータ
- 2F ビジネスパソコン
  - ●パソコン●ディスプレイ●プリンタ●専門書籍●パソコンアクセサリー
- 1F ホビーのパソコン
  - ●ホビーパソコン●MSX●ゲームソフト●学習ソフト



Personal Computer Store
J&P 渋谷店
東京都渋谷区道玄坂2丁目28番4号(〒150)
☎(03)496-4141〈水曜定休〉

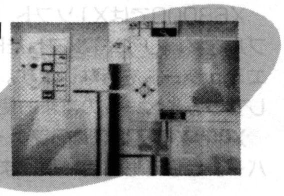
■Xホビーソフト

注文No.	タイトル	ソフトハウス	適応機種	メディア	価格
X7-146	リバイバー	アルシスソフト	MZ-2500	3.5″DD	¥6,800
X7-147	ウィバーン	アルシスソフト	MZ-2500	3.5″DD	¥6,800
X7-148	殺人クラブ	リバーヒル	MZ-2500	3.5″DD	¥7,800
X7-149	ドルアーガの塔	テンパ	MZ-2500	3.5″DD	¥6,800
X7-150	スペースハリアア	電波新聞社	X68000	5″2D	¥6,800
X7-151	ゼビウス	テンパ	X68000	5″2HD	¥6,800
X7-152	ザ・コックピット	コムパック	X68000	5″2HD	¥6,800
X7-153	上海	システムソフト	X68000	5″2HD	¥6,500
X7-154	アルカノイド	シャープ	X68000	5″2HD	¥7,800

X-68000対応コーナー

Z'sSTAFF PRO 68K

X7-155

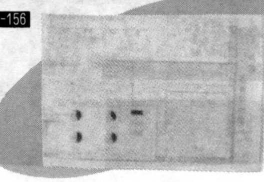


¥58,000 •ソフトハウス(ツァイト)

表現力の素晴しさに加えて、編集機能もPRO仕様。複雑なカラーチェンジから、モザイク変換、ソフトフォーカスまで、じっくりと手の込んだ作品を描くことが可能である。

超高性能●統合型スプレッドシート Kamikaze(神風)

X7-156



¥68,000 •ソフトハウス(サムシンググッド)

〈特長〉
- ●一度に16個までウィンドウをオープンできます。
- ●マウス完全対応の簡単なオペレーション。
- ●Kamikaza(神風)はワープロ以上の表現力を持ちます。
- ●簡単にデータをグラフ化することができます。

■ビジネスソフト

注文No.	タイトル	ソフトハウス	適応機種	メディア	価格
X7-165	C compiler	シャープ	X68000	5″2D	¥39,800
X7-166	MUSIC	シャープ	X68000	5″2D	¥15,800
X7-167	サウンドPRO 68K	シャープ	X68000	5″2D	¥15,800
X7-168	LINKS 68K	シャープ	X68000	5″2D	¥19,800
X7-169	日本語MY CARD・X1t	アバロン	X68000	5″2D (2)	¥58,000
X7-170	ビジレスIII	OAテック	X68000	5″2D	¥68,000
X7-171	HuCAL 日本語	ハドソン	X68000	5″2D	¥45,000
X7-172	Multiplan	シャープ	X68000	5″2D (2)	¥49,000

SUPER春望II X7-157

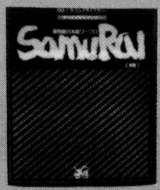
適応機種	X-1ターボ
ソフトハウス	テーピーソフト



グラフィック機能も内蔵日本語ワープロソフト
(3.5″D) **¥34,800**

高性能日本語ワープロ 即戦力Samurai(侍) X7-158

適応機種	X-1/X-1ターボ
ソフトハウス	サムシンググッド

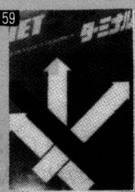


基本性能重視の日本語ワープロソフト
(5″2D) **¥19,800**

JETターボターミナル

適応機種	X-1ターボ
ソフトハウス	エス・ピー・エス

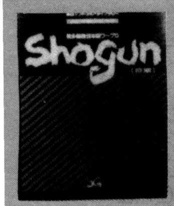
X7-159



オートログイン・オートダイヤル機能高性能通信ソフト
(5″2D) **¥9,800**

日本語ワープロ「将軍」 X7-160

適応機種	X-1ターボ
ソフトハウス	シャープ



性能ワープロソフト8ビット最高峰
″2D) **¥34,800**

Inkpot X7-161

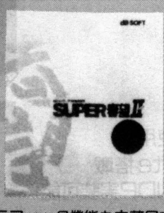
適応機種	X-1ターボ
ソフトハウス	アスキー



マウスでお絵かき、グラフィックソフト
(5″2D) **¥20,000**

SUPER春望II X7-162

適応機種	MZ-2500
ソフトハウス	テーピーソフト



グラフィック機能も内蔵日本語ワープロソフト
(3.5″D) **¥34,800**

Win DEX X7-163

適応機種	X-1ターボ
ソフトハウス	ジェー・イー・エル



プロフェッショナルマルチウインドウエディタ
(5″2D) **¥28,000**

プリントショップ X7-164

適応機種	X-1ターボ
ソフトハウス	ブローダバンドジャパン



楽しい印刷ソフト
(5″2D) **¥12,800**

お申し込み方法

右の注文書にご希望商品の注文No.および必要事項ご記入の上、現金書留にて J&P 渋谷店までお申し込みください。現金受領後、発送また、J&P HOTLINE会員の方は、ショッピングコーナーでもお申し込みいただけます。

記載以外のソフトのご注文も承りますので、詳しくはお電話にてお問い合わせ下さい。 ☎(03)496-4141

キリトリ線

現金書留申込み用紙

おところ 〒□□□□□

TEL ( )

おなまえ

様

注文No.(ディスクサイズ)	数量	金額
X7- ( )	本	円
X7- ( )	本	円
X7- ( )	本	円
合計	本	円
お手持の機種名		

お申込み先：東京都渋谷区道玄坂2丁目28番4号(〒150) J&P 渋谷店メールショッピング係

料金受取人払

郵便はがき

1 0 2

麴町局承認

7587

差出有効期間
昭和63年12月
31日まで

(受取人)

東京都千代田区
九段南2-3-26井関ビル

㈱日本ソフトバンク

Oh!X 編集室行

□□□-□□ 電話

住所

氏名 年齢

職業・勤務先
学校・学部・学年

●編集室へのメッセージ

今月号の特集について

いちばん良かった記事	興味のなかった記事
これから載せてほしい記事内容	本誌以外にお読みのパソコン雑誌

推薦する市販ソフト
ソフト名：
推薦理由：

あなたの愛機は（該当機種に○印をつけてください）　　　ない
X1(マニアタイプ, C, D, F, G, twin) X1turbo（model 10, 20, 30, 40, II, III, Z, ZII）
MZ-(80K／C, 1200, 700, 1500, 80B, 2000, 2200, 2500, 2861)
X68000(600, 601, 611)　　　その他
FD(　基)　TAPE　QD　HD(　　MB)　プリンタ(　　　　　　　　　)

年齢　　歳　　パソコン歴　　年　　男・女　　プレゼントNo.

SHARP New-Life People



NEW Z-BASIC搭載

多色グラフィック、カラー画像デジタイズ、ステレオFM音源、バンクメモリ対応などクリエイティブワークを強力にサポートするAV指向の高水準BASICです。グラフィック用関数、X68000と命令コンパチの拡張MMLをはじめ使い込むほどに凄さがわかるパワフルなBASICを搭載しました。

先駆のAVアート機能

量子化、モザイク、反転などトリック取り込み処理をサポートしたカラー画像デジタイズ機能標準装備。さらに、クロマキー合成、インターレーススーパーインポーズ、4,096色対応ニューテロッパ機能、8重和音のステレオFM音源。先駆のZアビリティがパソコンクリエイターを魅了します。

●メインメモリ128KB標準実装（NEW Z-BASICで最大576Kバイトまでサポート）した大容量設計 ●1Mバイトフロッピー2基搭載 ●JIS第1/第2水準準拠漢字ROM、「システム・ユーザー辞書」標準装備 ●マウス標準装備 ●X1ターボシリーズの豊富なソフト資産が活用できるコンパチブル設計 ●多彩な通信ツール※のサポートでパソコン通信に対応 ●ドットピッチ0.31mmの高精細カラーディスプレイテレビ※ ※別売

X1 パソコンテレビ turbo ZII

本体＋キーボード	CZ-881C-BK（ブラック）	標準価格 179,800円
14型カラーディスプレイテレビ	CZ-880D-BK（ブラック）	標準価格 109,800円
14型カラーディスプレイテレビ	CZ-830D-BK（ブラック）	標準価格 98,000円
チルトスタンド	CZ-6ST1-B（ブラック）	標準価格 5,800円

※写真のディスプレイはCZ-880Dです。

資料請求券
X1 turbo ZII
oh!X
7係

シャープ株式会社 ●お問い合わせは…シャープ㈱電子機器事業本部システム機器営業部〒545 大阪市阿倍野区長池町22番22号☎(06)621-1221(大代表)
電子機器事業本部テレビ事業部第4商品企画部〒162 東京都新宿区市谷八幡町8番地☎(03)260-1161(大代表)

T4910217907541 雑誌 02179-7

Oh!X 7月号 昭和63年7月1日発行（毎月1回1日発行）通巻第75号 昭和58年11月2日第三種郵便物認可 ㈱日本ソフトバンク発行 Printed in Japan 定価540円

Oh!X 1988 7 ■特集 実践C言語からの誘惑 ■速報 OS-9/X68000 ■熱烈歓迎 ゲームレビュー 日本ソフトバンク